OS/390

MVS JCL User's Guide

GC28-1758-05

OS/390

MVS JCL User's Guide

GC28-1758-05

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xiii.

Sixth Edition, March 1999

This is a major revision of GC28-1758-04.

This edition applies to Version 2 Release 7 of OS/390 (5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+914+432-9405
FAX (Other Countries):
Your International Access Code +1+914+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/s390/os390/

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1988, 1999. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices . . . xiii
Trademarks . . . xiii

About This Book . . . xv
Who Should Use This Book . . . xv
Where to Find More Information . . . xv
Programs . . . xv
Hardware . . . xvi

Summary of Changes . . . xvii

Part 1. Introduction

Chapter 1. Introduction - Job Control Statements . . . 1-1

Chapter 2. Introduction - Job Control Language (JCL) . . . 2-1
Understanding JCL . . . 2-1
"Chez MVS" . . . 2-1
How This Relates to JCL . . . 2-1
Job Control Statements . . . 2-3
Required Control Statements . . . 2-3
Exercise: Creating and Entering a Job . . . 2-3
Before You Begin . . . 2-3
Step 1. Allocate a Data Set to Contain Your JCL . . . 2-5
Step 2. Edit the JCL Data Set and Add the Necessary JCL . . . 2-5
Step 3. Submit the JCL to the System as a Job . . . 2-6
Step 4. View and Understand the Output from the Job . . . 2-8
Step 5. Make Changes to Your JCL . . . 2-9
Step 6. View and Understand Your Final Output . . . 2-10
More Complex Jobs . . . 2-12
In-Stream and Cataloged Procedures . . . 2-12
Input Streams . . . 2-13
Additional Information . . . 2-14
Installation Conventions Worksheet . . . 2-14
Using ISPF to Allocate and Edit a Data Set . . . 2-15
Using SDSF to View Held Output from a Job . . . 2-17
Helpful Utilities . . . 2-19

Chapter 3. Job Control Tasks . . . 3-1
Entering Jobs . . . 3-1
Processing Jobs . . . 3-2
Requesting Resources . . . 3-2
Task Charts . . . 3-2

Part 2. Tasks for Entering Jobs

Chapter 4. Entering Jobs - Identification . . . 4-1
Identification of Job . . . 4-1
Identification of Step . . . 4-2

© Copyright IBM Corp. 1988, 1999

Identification of Procedure 4-2
Identification of INCLUDE Group 4-3
Identification of Account 4-3
For Local Execution 4-3
For Remote Execution 4-4
Identification of Programmer 4-4

Chapter 5. Entering Jobs - Execution 5-1
Execution of Program 5-1
Execution of Procedure 5-2
Execution when Restarting and with Checkpointing (non-APPC) 5-2
Restarting after Abnormal Termination 5-2
Restarting When the System Failed in a JES2 System 5-3
Restarting When the System Failed in a JES3 System 5-3
Deadline or Periodic Execution in a JES3 System 5-4
Use of Deadline Scheduling 5-4
Use of Periodic Scheduling 5-4
Execution when Dependent on Other Jobs in a JES3 System 5-5
Execution at Remote Node (non-APPC) 5-7
Considerations when Submitting a Remote Job 5-8

Chapter 6. Entering Jobs - Job Input Control 6-1
Job Input Control by Holding Job Entrance (Non-APPC) 6-1
Job Input Control by Holding Local Input Reader (Non-APPC) 6-2
Job Input Control by Copying Input Stream (Non-APPC) 6-2
Job Input Control from Remote Work Station 6-3
JES2 Remote Job Entry 6-3
JES3 Remote Job Processing 6-4

Chapter 7. Entering Jobs - Communication 7-1
Communication from JCL to System (Non-APPC) 7-1
Communication from JCL to Operator (Non-APPC) 7-2
Communication from JCL to Programmer 7-2
Communication from JCL to Program 7-3
PARM Values for IBM-Supplied Programs 7-3
Communication from System to Operator 7-3
Messages during Volume Mounting 7-3
Messages When Job Exceeds Output Limit 7-4
Communication from System to Userid 7-5
Job Completion 7-5
Print Completion 7-6
Communication from Time Sharing Userid to a JES3 System 7-6
Communication from Functional Subsystem to Programmer 7-6
Communication through Job Log 7-6
Printing Job Log and Sysout Data Sets Together 7-8

Chapter 8. Entering Jobs - Protection 8-1
Protection through RACF 8-1

Chapter 9. Entering Jobs - Resource Control 9-1
Resource Control of Program Library 9-1
System Library 9-1
Private Library 9-2
Temporary Library 9-4

Resource Control of Procedure Library 9-5
Retrieving a Procedure Library 9-5
Updating a Procedure Library 9-5
Resource Control of INCLUDE Group 9-6
Retrieving an INCLUDE Group 9-6
Resource Control of Address Space 9-6
Types of Storage 9-7
Requesting Amount and Type of Storage 9-7
Resource Control of the Processor 9-9
Selecting a Processor Using A Scheduling Environment 9-9
Selecting a Processor in JES2 9-10
Selecting a Processor in JES3 9-10
Resource Control of Spool Partitions in a JES3 System 9-11

Part 3. Tasks for Processing Jobs

Chapter 10. Processing Jobs - Processing Control 10-1
Processing Control by Conditional Execution 10-1
Bypassing or Executing Steps Based on the Evaluation of Previous Steps 10-1
Bypassing or Executing Steps Based on Return Codes 10-6
Processing Control by Cancelling a Job that Exceeds Output Limit 10-13
Limiting Output in an APPC Scheduling Environment 10-13
Limiting Output in a Non-APPC Scheduling Environment 10-14
Use in Testing 10-14
Processing Control by Timing Execution 10-15
JOB and EXEC TIME Parameter 10-15
JES2 Time Parameters 10-16
OS/390 UNIX System Services Considerations 10-16
Processing Control for Testing 10-16
Altering Usual Processing for Testing 10-16

Chapter 11. Processing Jobs - Performance Control 11-1
Performance Control by Job Class Assignment (Non-APPC) 11-1
Performance Control by Selection Priority (Non-APPC) 11-2
Priority for JES2 Jobs 11-2
Priority for JES3 Jobs 11-3
Priority Aging 11-3
Performance Control by Performance Group (Non-APPC) 11-3
Performance Control by I/O-to-Processing Ratio (Non-APPC) 11-4

Part 4. Tasks for Requesting Data Set Resources

Chapter 12. Data Set Resources - Identification 12-1
Identification of Data Set 12-1
Permanent Data Set 12-1
Temporary Data Sets 12-3
Copying the Data Set Name from an Earlier DD Statement 12-5
Concatenating Data Sets 12-5
Identification of In-Stream Data Set (Non-APPC) 12-5
Entering Data Through the Input Stream 12-5
In-Stream Data Sets in a JES3 System 12-6
Identification of Data Set on 3540 Diskette Input/Output Unit 12-7

Identification through Catalog 12-7
Using Private Catalogs 12-8
Identification through Label 12-8
Identification by Location on Tape 12-10
Identification as TCAM Message Data Set 12-10
Identification as Data Set from or to Terminal (Non-APPC) 12-11

Chapter 13. Data Set Resources - Description 13-1
Description of Status 13-1
Data Set Integrity Processing 13-2
Description of Data Attributes 13-5
In Data Control Block (DCB) 13-5
Migration and Backup (with SMS) 13-7

Chapter 14. Data Set Resources - Protection 14-1
Protection through RACF 14-1
Protection with the PROTECT Parameter 14-1
Protection with the SECMODEL Parameter 14-2
Protection for ISO/ANSI/FIPS Version 3 Tapes 14-2
Protection by Passwords 14-3
Protection of Access to BSAM or BDAM Data Sets 14-3

Chapter 15. Data Set Resources - Allocation 15-1
Allocation of Device 15-2
Device Allocation for SMS-Managed Data Sets 15-2
Device Allocation for Non-SMS-Managed Data Sets 15-3
Device Allocation in a JES3 System 15-12
Allocation of Volume 15-16
Volume Allocation for SMS-Managed Data Sets 15-17
Volume Allocation for Non-SMS-Managed Data Sets 15-18
Volume Allocation for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume 15-18
Interactions Between Device and Volume Allocation 15-24
Relationship of the UNIT and VOLUME Parameters (Non-SMS-Managed Data Sets) 15-24
Relationship of the UNIT and VOLUME Parameters (SMS-Managed Data Sets) 15-29
Unit and Volume Affinity for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume 15-30
Stacking Data Sets 15-39
Examples of Data Set Stacking 15-40
Data Set Stacking and TMM 15-42
Allocation of Direct Access Space 15-44
Requesting System Assigned Space 15-44
Requesting Specific Tracks 15-49
Allocation of Virtual I/O 15-49
Backward References to VIO Data Sets 15-51
Allocation with Volume Premounting in a JES2 System 15-52
Dynamic Allocation 15-53

Chapter 16. Data Set Resources - Processing Control 16-1
Processing Control by Suppressing Processing 16-1
Processing Control by Postponing Specification 16-2
Processing Control with Checkpointing 16-4

Processing Control by Subsystem 16-4
Requesting Subsystem 16-4
Program Control Statements for a Subsystem 16-5
Processing Control by TCAM Job or Task 16-5

Chapter 17. Data Set Resources - End Processing 17-1
Unallocation End Processing 17-1
Disposition End Processing of Data Set 17-1
Disposition Controlled by DISP Parameter 17-2
Disposition Controlled by Time 17-11
Release of Unused Direct Access Space in End Processing 17-12
Disposition End Processing of Volume 17-12
Disposition of Removable Volumes 17-13
Volume Retention 17-13

Part 5. Tasks for Requesting Sysout Data Set Resources

Chapter 18. Sysout Resources - Identification 18-1
Identification as a Sysout Data Set 18-1
Identification of Output Class 18-2
Identification of Data Set on 3540 Diskette Input/Output Unit 18-2

Chapter 19. Sysout Resources - Description 19-1
Description of Data Attributes 19-1

Chapter 20. Sysout Resources - Protection 20-1
Protection of Printed Output 20-1

Chapter 21. Sysout Resources - Performance Control 21-1
Performance Control by Queue Selection (non-APPC) 21-1

Chapter 22. Sysout Resources - Processing Control 22-1
Processing Control with Additional Parameters 22-1
Adding Parameters from OUTPUT JCL Statement 22-2
Adding Parameters from JES2 /*OUTPUT Statement 22-4
Adding Parameters from JES3 //*FORMAT Statement 22-4
Processing Control by Segmenting 22-5
Processing Control with Other Data Sets 22-5
Using Output Class 22-5
Using Sysout Data Set Size in a JES3 System 22-5
Using Groups in a JES2 System 22-6
Processing Control by External Writer 22-7
Processing Control by Mode 22-7
Processing Control by Holding 22-8
Holding Using the DD Statement 22-8
Holding Using the OUTPUT JCL Statement 22-8
Processing Control by Suppressing Output 22-9
Using Dummy Status to Suppress Output 22-9
Using Class to Suppress Output in a JES2 System 22-10
Using the OUTPUT JCL Statement to Suppress Output in a JES2 System 22-10
Processing Control with Checkpointing 22-10
Processing Control by Print Services Facility 22-11
Identifying a Library to PSF 22-11

Chapter 23. Sysout Resources - End Processing 23-1
Unallocation End Processing 23-1

Chapter 24. Sysout Resources - Destination Control 24-1
Destination Control to Local or Remote Device or to Another Node 24-1
Multiple Destinations 24-2
Controlling Output Destination in a JES2 Network 24-2
Controlling Output Destination in a JES3 Network 24-5
Destination Control to Another Processor in a JES3 System 24-6
Destination Control to Internal Reader 24-6
Destination Control to Terminal 24-7
Destination Control to Assist in Sysout Distribution 24-8

Chapter 25. Sysout Resources - Output Formatting 25-1
Output Formatting to Any Printer 25-1
Output Formatting to 3800 Printing Subsystem 25-2
Copy Modification 25-3
Character Arrangements 25-3
Output Formatting to 3211 Printer with Indexing Feature in a JES2 System . 25-4
Output Formatting to Punch 25-5
Interpretation of Punched Cards 25-5
Output Formatting of Dumps on 3800 Printing Subsystem 25-5

Chapter 26. Sysout Resources - Output Limiting 26-1
Output Limiting 26-1
Limiting Output in an APPC Scheduling Environment 26-1
Limiting Output in a Non-APPC Scheduling Environment 26-2
Actions when Limit Exceeded 26-2

Chapter 27. Sysout Resources - USERDATA OUTPUT JCL Keyword . . 27-1
References 27-1
Examples 27-1

Part 6. Examples

Chapter 28. Example - Assemble, Linkedit, and Go 28-1

Chapter 29. Example - Multiple Output 29-1

Chapter 30. Example - Obtaining Output in a JES2 System 30-1

Chapter 31. Example - Obtaining Output in a JES3 System 31-1

Chapter 32. Example - Identifying Data Sets to the System 32-1

Appendixes

Appendix A. Indexed Sequential Data Sets A-1
Creating an Indexed Sequential Data Set A-1
Procedure when Allocation Error Occurs A-4
Area Arrangement of an Indexed Sequential Data Set A-5
Retrieving an Indexed Sequential Data Set A-5

Appendix B. Generation Data Sets B-1
Building a GDG Base Entry B-2
Defining Attributes for SMS-Managed Generation Data Sets B-2
Creating an SMS-Managed Generation Data Set B-3
Disposition of SMS-Managed Generation Data Sets B-3
Defining Attributes for Non-SMS-Managed Generation Data Sets B-4
Creating a Non-SMS-Managed Generation Data Set B-5
Retrieving a Generation Data Set B-7
Deleting and Uncataloging Generation Data Sets B-9
Submitting a Job for Restart B-10

Appendix C. VSAM Data Sets C-1
VSAM Data Sets - With SMS C-1
Creating a VSAM Data Set - With SMS C-1
Retrieving an Existing VSAM Data Set - With SMS C-1
Migration Consideration for SMS C-1
DD Statement Parameters - With SMS C-2
VSAM Data Sets - Without SMS C-4
Creating a VSAM Data Set - Without SMS C-4
Retrieving an Existing VSAM Data Set - Without SMS C-4
DD Statement Parameters - Without SMS C-4

Appendix D. Data Sets with SMS D-1
SMS Constructs D-1
Existing JCL D-2
Default Unit D-2
Specifying Constructs D-2
Overriding Attributes Defined in the Data Class D-3
Overriding Attributes Defined in the Management Class D-3
Overriding Attributes Defined in the Storage Class D-4
Protecting Data Sets with RACF D-4
Modeling Data Set Attributes D-4

Index X-1

Figures

1-1. Job Control Statements . . . 1-1
2-1. JCL-Related Actions (User and MVS) . . . 2-2
2-2. Output from Job Invoking IEFBR14 Program . . . 2-9
2-3. Output from Job Invoking SORT Program . . . 2-11
2-4. In-Stream Procedure . . . 2-12
2-5. Cataloged Procedure: Member MYPROC in SYS1.PROCLIB . . . 2-13
2-6. Job that Executes Cataloged Procedure MYPROC . . . 2-13
2-7. Job Boundaries in a Three-Job Input Stream . . . 2-14
2-8. Tasks and Utility Programs . . . 2-20
3-1. Tasks for Entering Jobs . . . 3-3
3-2. Tasks for Processing Jobs . . . 3-5
3-3. Tasks for Requesting Data Set Resources . . . 3-6
3-4. Tasks for Requesting Sysout Data Set Resources . . . 3-8
4-1. Identification Task for Entering Jobs . . . 4-1
5-1. Execution Task for Entering Jobs . . . 5-1
6-1. Input Control Task for Entering Jobs . . . 6-1
7-1. Communication Task for Entering Jobs . . . 7-1
8-1. Protection Task for Entering Jobs . . . 8-1
9-1. Resource Control Task for Entering Jobs . . . 9-1
10-1. Processing Control Task for Processing Jobs . . . 10-1
11-1. Performance Control Task for Processing Jobs . . . 11-1
12-1. Identification Task for Requesting Data Set Resources . . . 12-1
13-1. Description Task for Requesting Data Set Resources . . . 13-1
13-2. Data Set Integrity Processing . . . 13-4
14-1. Protection Task for Requesting Data Set Resources . . . 14-1
14-2. Processing with DD LABEL Subparameter IN or OUT . . . 14-4
15-1. Allocation Task for Requesting Data Set Resources . . . 15-1
15-2. Effect of Device Status on Allocation . . . 15-3
15-3. JES3 Job Setup (SETUP=JOB) . . . 15-14
15-4. JES3 High Watermark Setup (SETUP=HWS) . . . 15-15
15-5. JES3 Explicit Setup (SETUP=ddname) . . . 15-16
15-6. Unit-Affinity Examples of Tape Library Requests . . . 15-32
15-7. Unit and Volume Affinity (Non-SMS-Managed Data Sets) . . . 15-34
15-8. IBM-Recommended Parameters for Data Set Stacking . . . 15-39
16-1. Processing Control Task for Requesting Data Set Resources . . . 16-1
17-1. End Processing Task for Requesting Data Set Resources . . . 17-1
18-1. Identification Task for Requesting Sysout Data Set Resources . . . 18-1
19-1. Description Task for Requesting Sysout Data Set Resources . . . 19-1
20-1. Protection Task for Requesting Sysout Data Set Resources . . . 20-1
21-1. Performance Control Task for Requesting Sysout Data Set Resources . . . 21-1
22-1. Processing Control Task for Requesting Sysout Data Set Resources 22-1
23-1. End Processing Task for Requesting Sysout Data Set Resources . 23-1
24-1. Destination Control Task for Requesting Sysout Data Set Resources 24-1
25-1. Output Formatting Task for Requesting Sysout Data Set Resources 25-1
26-1. Output Limiting Task for Requesting Sysout Data Set Resources . 26-1
A-1. Area Arrangement of ISAM Data Sets . . . A-5
A-2. DD Parameters for Retrieving or Extending an ISAM Data Set . . . A-7
C-1. With SMS, DD Parameters to Use when Processing VSAM Data Sets C-2

© Copyright IBM Corp. 1988, 1999

C-2. With SMS, DD Parameters to Avoid when Processing VSAM Data Sets C-3
C-3. Without SMS, DD Parameters to Use when Processing VSAM Data Sets C-5
C-4. Without SMS, DD Parameters to Avoid when Processing VSAM Data Sets C-6

Notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

| IBM Director of Licensing
| IBM Corporation
| North Castle Drive
| Armonk, NY 10504-1785
| USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- AFP
- DB2
- DFSMSdfp
- DFSMS/MVS
- Hardware Configuration Definition
- IBM
- IBMLink
- MVS/ESA
- MVS/SP
- OpenEdition

© Copyright IBM Corp. 1988, 1999

- OS/390
- Print Services Facility
- PSF
- RACF
- SP
- System/390
- System/370
- VTAM

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

About This Book

This book describes the job control tasks needed to enter jobs into the operating system, control the system's processing of jobs, and request the resources needed to run jobs. To perform the tasks, programmers code job control statements. This book describes how to use these statements, which consist of:

- Job control language (JCL) statements
- Job entry subsystem 2 (JES2) control statements
- Job entry subsystem 3 (JES3) control statements

This book is designed as a user's guide, to be used when deciding how to perform job control tasks. It does not describe how to code the statements. For an introduction to the statements and for coding information, see the companion book, *OS/390 MVS JCL Reference*, GC28-1757.

Who Should Use This Book

This book is for system and application programmers who enter programs into the operating system. Those using this book should understand the concepts of job management and data management.

Where to Find More Information

To have complete JCL information, you need the following book:

OS/390 MVS JCL Reference, GC28-1757

Where necessary, this book references information in other books, using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of OS/390, see *OS/390 Information Roadmap*, GC28-1727. The following tables list titles and order numbers for books related to other products.

Programs

Short Title Used in This Book	Title	Order Number
ACF/TCAM Installation Reference	*Advanced Communications Function for TCAM, Version 2 Installation Reference*	SC30-3133
ISPF/PDF Guide and Reference	*ISPF/PDF Guide and Reference V3.4 for MVS*	SC34-4258
PSF/MVS System Programming Guide	*PSF/MVS System Programming Guide*	S544-3672
PSF/MVS Application Programming Guide	*PSF/MVS Application Programming Guide*	S544-3673

© Copyright IBM Corp. 1988, 1999

Hardware

Short Title Used in This Book	Title	Order Number
2821 Component Description	*IBM 2821 Control Unit Component Description*	GA24-3312
None	*IBM 3340 Disk/Storage - Fixed Head Feature User's Guide*	GA26-1632
3540 Programmer's Reference	*OS/VS2 IBM 3540 Programmer's Reference*	GC24-5111
3800 Programmer's Guide	*IBM 3800 Printing Subsystem Programmer's Guide*	GC26-3846
Forms Design Reference Guide for the 3800	*Forms Design Reference Guide for the IBM 3800 Printing Subsystem*	GA26-1633

Summary of Changes

Summary of Changes
for GC28-1758-05
OS/390 Version 2 Release 7

The book contains information previously presented in GC28-1758-04, which supports OS/390 Version 1 Release 6.

New Information

- The CCSID parameter is added to the JOB, EXEC, and DD statements.
- The topic "Data Sets That Span Libraries" is added to Chapter 15, “Data Set Resources - Allocation” on page 15-1 to describe an enhancement to allocation processing.

Changed Information

- The protection task of the ACCODE parameter of the DD statement is updated.

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
for GC28-1758-04
OS/390 Version 2 Release 6

The book contains information previously presented in GC28-1758-03, which supports OS/390 Version 1 Release 5.

Changed Information

- The SCHENV parameter on the JOB statement, which allows you to specify the name of a WLM scheduling environment, can now be used for JES3 jobs.
- As part of the name change of OpenEdition to OS/390 UNIX System Services, occurrences of OS/390 OpenEdition have been changed to OS/390 UNIX System Services or its abbreviated name, OS/390 UNIX. OpenEdition may continue to appear in messages, panel text, and other code with OS/390 UNIX System Services.

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
for GC28-1758-03
OS/390 Version 2 Release 5

The book contains information previously presented in GC28-1758-02, which supports OS/390 Version 1 Release 4.

New Information

© Copyright IBM Corp. 1988, 1999

- New OUTPUT JCL parameters have been added to the Print Services Facility (PSF).

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
for GC28-1758-02
OS/390 Version 2 Release 4

The book contains information previously presented in GC28-1758-01, which supports OS/390 Version 1 Release 2.

New Information

- There is a new parameter, SCHENV, on the JOB statement. Currently valid only for JES2 jobs, it allows you to specify the name of a WLM scheduling environment.

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
for GC28-1758-01
OS/390 Release 2

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
for GC28-1758-00
OS/390 Release 1

This book contains information previously presented in *MVS/ESA JCL User's Guide* GC28-1473, which supports MVS/ESA System Product Version 5.

This book includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Part 1. Introduction

Chapter 1. Introduction - Job Control Statements 1-1

Chapter 2. Introduction - Job Control Language (JCL) 2-1
Understanding JCL 2-1
"Chez MVS" 2-1
How This Relates to JCL 2-1
Job Control Statements 2-3
Required Control Statements 2-3
Exercise: Creating and Entering a Job 2-3
Before You Begin 2-3
Step 1. Allocate a Data Set to Contain Your JCL 2-5
Step 2. Edit the JCL Data Set and Add the Necessary JCL 2-5
Step 3. Submit the JCL to the System as a Job 2-6
Step 4. View and Understand the Output from the Job 2-8
Step 5. Make Changes to Your JCL 2-9
Step 6. View and Understand Your Final Output 2-10
More Complex Jobs 2-12
In-Stream and Cataloged Procedures 2-12
In-Stream Procedures 2-12
Cataloged Procedures 2-12
Input Streams 2-13
Additional Information 2-14
Installation Conventions Worksheet 2-14
Using ISPF to Allocate and Edit a Data Set 2-15
Using SDSF to View Held Output from a Job 2-17
Helpful Utilities 2-19

Chapter 3. Job Control Tasks 3-1
Entering Jobs 3-1
Processing Jobs 3-2
Requesting Resources 3-2
Task Charts 3-2

For your program to execute on the computer and perform the work you designed it to do, your program must be processed by your operating system. Your operating system consists of a base control program (BCP) with a job entry subsystem (JES2 or JES3) and DFSMSdfp installed with it.

For the operating system to process a program, programmers must perform certain job control tasks. These tasks are performed through the job control statements, which are listed in the first chapter. The job control tasks are introduced in the second chapter as well as introductory information about JCL. The charts in the third chapter divide these tasks into detailed subtasks. The tasks are:

- Entering jobs
- Processing jobs
- Requesting resources

© Copyright IBM Corp. 1988, 1999

Part 1. Introduction

Chapter 1. Introduction - Job Control Statements

This chapter lists in Figure 1-1 all the job control statements and gives the purpose of each statement.

Figure 1-1 (Page 1 of 3). Job Control Statements

Statement	Name	Purpose
JCL Statements		
// command	JCL command	Enters an MVS system operator command through the input stream. The command statement is used primarily by the operator. Use the COMMAND statement instead of the JCL command statement.
// COMMAND	command	Specifies an MVS or JES command that the system issues when the JCL is converted. Use the COMMAND statement instead of the JCL command statement.
//* comment	comment	Contains comments. The comment statement is used primarily to document a program and its resource requirements.
// CNTL	control	Marks the beginning of one or more program control statements.
// DD	data definition	Identifies and describes a data set.
/*	delimiter	Indicates the end of data placed in the input stream. **Note:** Any two characters can be designated by the user to be the delimiter.
// ENDCNTL	end control	Marks the end of one or more program control statements.
// EXEC	execute	Marks the beginning of a job step; assigns a name to the step; identifies the program or the cataloged or in-stream procedure to be executed in this step.
// IF/THEN/ELSE/ENDIF	IF/THEN/ELSE/ENDIF statement construct	Specifies conditional execution of job steps within a job.
// INCLUDE	include	Identifies a member of a partitioned data set (PDS) or partitioned data set extended (PDSE) that contains JCL statements to be included in the job stream.

© Copyright IBM Corp. 1988, 1999

Figure 1-1 (Page 2 of 3). Job Control Statements		
// JCLLIB	JCL library	Identifies the libraries that the system will search for: • INCLUDE groups • Procedures named in EXEC statements.
// JOB	job	Marks the beginning of a job; assigns a name to the job.
//	null	Marks the end of a job.
// OUTPUT	output JCL	Specifies the processing options that the job entry subsystem is to use for printing a sysout data set.
// PEND	procedure end	Marks the end of an in-stream or cataloged procedure.
// PROC	procedure	Marks the beginning of an in-stream procedure and may mark the beginning of a cataloged procedure; assigns default values to parameters defined in the procedure.
// SET	set	Defines and assigns initial values to symbolic parameters used when processing JCL statements. Changes or nullifies the values assigned to symbolic parameters.
// XMIT	transmit	Transmits input stream records from one node to another. **Note:** The XMIT JCL statement is supported only on JES3 systems.
Statement	**Purpose**	
JES2 Control Statements		
/*$command	Enters JES2 operator commands through the input stream.	
/*JOBPARM	Specifies certain job-related parameters at input time.	
/*MESSAGE	Sends messages to the operator via the operator console.	
/*NETACCT	Specifies an account number for a network job.	
/*NOTIFY	Specifies the destination of notification messages.	
/*OUTPUT	Specifies processing options for sysout data set(s).	
/*PRIORITY	Assigns a job queue selection priority.	
/*ROUTE	Specifies the output destination or the execution node for the job.	
/*SETUP	Requests mounting of volumes needed for the job.	
/*SIGNOFF	Ends a remote job stream processing session.	
/*SIGNON	Begins a remote job stream processing session.	
/*XEQ	Specifies the execution node for a job.	
/*XMIT	Indicates a job or data stream to be transmitted to another JES2 node or eligible non-JES2 node.	
Statement	**Purpose**	

Figure 1-1 (Page 3 of 3). Job Control Statements	
JES3 Control Statements	
//**command	Enters JES3 operator commands, except *DUMP and *RETURN, through the input stream.
//*DATASET	Begins an input data set in the input stream.
//*ENDDATASET	Ends the input data set that began with a //*DATASET statement.
//*ENDPROCESS	Ends a series of //*PROCESS statements.
//*FORMAT	Specifies the processing options for a sysout or JES3-managed print or punch data set.
//*MAIN	Defines selected processing parameters for a job.
//*NET	Identifies relationships between predecessor and successor jobs in a dependent job control net.
//*NETACCT	Specifies an account number for a network job.
//*OPERATOR	Sends messages to the operator.
//*PAUSE	Halts the input reader.
//*PROCESS	Identifies a nonstandard job.
//*ROUTE	Specifies the execution node for the job.
/*SIGNOFF	Ends a remote job stream processing session.
/*SIGNON	Begins a remote job stream processing session.

Chapter 2. Introduction - Job Control Language (JCL)

This chapter is divided into the following sections:

Heading	Description
"Understanding JCL"	Explains the purpose of JCL and how it is used.
"Exercise: Creating and Entering a Job" on page 2-3	Provides an example of JCL code that you can use as a basis for your own jobs.
"More Complex Jobs" on page 2-12	Explains how to create and use in-stream and cataloged procedures and how to group more than one job into input streams.
"Additional Information" on page 2-14	Contains a worksheet for documenting installation conventions; explains how to use ISPF to allocate and edit a data set; explains how to use SDSF to view held output from a job; and lists utilities that you can use with JCL to accomplish various tasks.

Understanding JCL

To get your MVS system to do work for you, you must describe to the system the work you want done and the resources you will need.

You use **Job Control Language (JCL)** to provide this information to MVS.

"Chez MVS"

One way of thinking about JCL is to compare it to a menu in a restaurant.

If you are a customer at a restaurant, you and the other customers don't just walk into the kitchen and start cooking your own dinners—that would defeat the very purpose of going to a restaurant. Instead, from a menu describing all the restaurant has to offer, you select items to make up an order, specifying which entrees you want, which salad dressing you prefer, and any other special requests you have. You then ask the waiter to take your order to the kitchen.

In the kitchen, a team of chefs divides up the work and the appropriate ingredients in order to prepare each dish as quickly and efficiently as possible. While the meals are being prepared, you and your friends can ignore what's going on in the kitchen, engaging instead in dinner conversation, catching up on the latest news. When the waiter brings your meal out, you concentrate on your enjoyment of the meal.

How This Relates to JCL

Now imagine yourself back at the office using your MVS system, and think of JCL as the menu. In the same way that you and the other diners select items from the menu and place orders for the waiter to take to the team of chefs, you and other MVS users use JCL to define work requests (called jobs), and use a job entry subsystem (JES) to submit those jobs to MVS.

© Copyright IBM Corp. 1988, 1999

Using the information that you and the other users provide with JCL statements, MVS allocates the resources needed to complete all of your jobs—just as the kitchen chefs divided up the work to prepare the orders of all the customers.

And just as the chefs worked in the kitchen while you and the other diners devoted your attention to what was going on at your tables, MVS completes the submitted jobs in the **background** of the system, enabling you and the other users to continue working on other activities in the **foreground**.

And just as the waiter conveys the results of the chefs' work to you, JES presents the output of the jobs to you.

Figure 2-1 shows an overview of the job-submission process. The user performs the parts on the left side of the figure, and the system performs the parts on the right. In this figure, MVS and JES comprise the “system”. Later in this introduction, distinctions will be made between MVS and JES, and between the two versions of JES (JES2 and JES3).

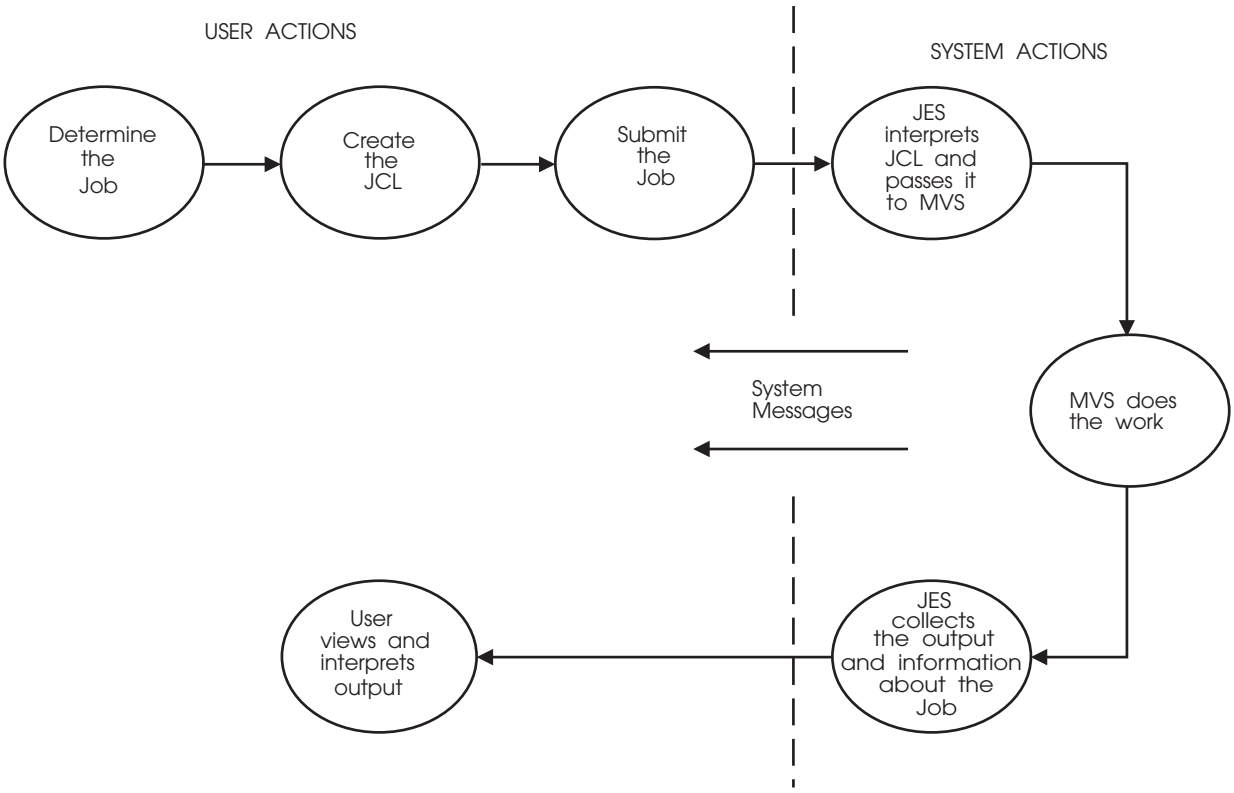

USER ACTIONS
SYSTEM ACTIONS
Determine the Job
Create the JCL
Submit the Job
JES interprets JCL and passes it to MVS
System Messages
MVS does the work
User views and interprets output
JES collects the output and information about the Job

Figure 2-1. JCL-Related Actions (User and MVS)

Job Control Statements

For every job that you submit, you need to tell MVS where to find the appropriate input, how to process that input (that is, what program or programs to run), and what to do with the resulting output.

You use JCL to convey this information to MVS through a set of statements known as **job control statements**. JCL's set of job control statements is quite large, enabling you to provide a great deal of information to MVS.

Most jobs, however, can be run using a very small subset of these control statements. Once you become familiar with the characteristics of the jobs you typically run, you may find that you need to know the details of only some of the control statements.

Within each job, the control statements are grouped into **job steps**. A job step consists of all the control statements needed to run *one* program. If a job needs to run more than one program, the job would contain a different job step for each of those programs.

Required Control Statements

Every job must contain a *minimum* of the following two types of control statements:

- *A JOB statement*, to mark the beginning of a job and assign a name to the job. The JOB statement is also used to provide certain administrative information, including security, accounting, and identification information. Every job has *one and only one* JOB statement.
- *An EXEC (execute) statement*, to mark the beginning of a job step, to assign a name to the step, and to identify the program or procedure to be executed in the step. You can add various parameters to the EXEC statement to customize the way the program executes. Every job has *at least one* EXEC statement.

In addition to the JOB and EXEC statements, most jobs usually also contain:

- *One or more DD (data definition) statements*, to identify and describe the input and output data to be used in the step. The DD statement may be used to request a previously-created data set, to define a new data set, to define a temporary data set, or to define and specify the characteristics of the output.

Chapter 1 lists the complete set of job control statements.

Exercise: Creating and Entering a Job

The following exercise shows you how to group the basic set of control statements into a job step within a job, how to submit your job, and how to understand the resulting output.

Before You Begin

Before creating any job, you need to know the following:

- ***Installation conventions.*** Every job must include special accounting and identifying information. However, the way this information is specified varies from one MVS installation to another.

In order to submit your JCL successfully, you need to find out the conventions that are followed at your installation.

A worksheet has been provided at the end of this chapter (see “Installation Conventions Worksheet” on page 2-14) as a guide for documenting this information. You may need to ask someone more familiar with your installation to help you identify the conventions indicated in the worksheet.

- ***How to allocate and edit a data set.*** During the exercise, you will be entering JCL statements into a data set so that you can subsequently modify and re-use them as required. Therefore, you must know how to use ISPF panels (or an equivalent technique) to allocate and edit the data set according to the specific requirements of your MVS system. See “Using ISPF to Allocate and Edit a Data Set” on page 2-15 for more information.

  Notes:

  1. It is a common programming practice to give any data set containing JCL a name that ends in JCL, such as *userid*.SORT.JCL.
  2. A data set that contains JCL must have a fixed-block format (RECFM=FB) with a logical record length of 80 (LRECL=80).

- ***The job to be done and the resources needed.*** You need to determine what work you plan to have MVS perform:

  – What inputs (resources) you will need and where they are located
  – What program you plan to use.
  – Where the output, if any, should go. (When the job completes, you will either dispose of the output or hold it for later printing or for viewing.)

  The job for this exercise is to sort a simple file and list the contents alphabetically. Decisions about inputs, outputs, and processing have already been made for you so that when you reach “Step 2. Edit the JCL Data Set and Add the Necessary JCL” on page 2-5, all you will have to do is to copy the example code provided.

- ***How to view and understand held output.*** Running your job will produce three types of held output:

  – System messages (JES and MVS)
  – Your JCL code with procedures expanded, overrides applied, and symbolics resolved.
  – Output as requested by the JCL code

  Held output may be viewed, printed, or purged. “Using SDSF to View Held Output from a Job” on page 2-17 explains how to use SDSF to view JCL output.

  In the example, “Step 4. View and Understand the Output from the Job” on page 2-8 and “Step 6. View and Understand Your Final Output” on page 2-10 show you how the output from the exercise should look and explain what each part of the output means.

Step 1. Allocate a Data Set to Contain Your JCL

Use ISPF (or equivalent function) to allocate a data set named *userid*.SORT.JCL (where *userid* is your TSO user ID) with a fixed-block format (RECFM=FB) and a logical record length of 80 (LRECL=80).

If you are not sure how to do this, see “Using ISPF to Allocate and Edit a Data Set” on page 2-15.

Step 2. Edit the JCL Data Set and Add the Necessary JCL

Use ISPF (or equivalent function) to edit the data set that you just allocated.

Enter the following JCL statements into the data set. Note that all JCL statements start with the special identifier //.

```
//SORT JOB 'accounting_data', 1
//  'user_name', 2
//  NOTIFY=&SYSUID, 3
//  MSGCLASS=message_class, 4
//  MSGLEVEL=(1,1), 5
//  CLASS=n, 6
//STEP1    EXEC  PGM=IEFBR14  7
//SORTIN   DD  *  8
NEPTUNE  9
PLUTO
EARTH
VENUS
MERCURY
MARS
URANUS
SATURN
JUPITER
/*  10
//SORTOUT  DD SYSOUT=*  11
/*  12
```

In the JCL code above:

1 Replace *accounting_data* with the appropriate security classification and identification information, according to the information you filled in on “Installation Conventions Worksheet” on page 2-14.

2 Replace *user_name* with your name.

3 NOTIFY= tells the system where to send “job complete” information. &SYSUID tells the system to automatically insert your user ID here, so the information will be sent to you.

4 MSGCLASS= tells the system what to do with messages the system sends you as it processes your job; for example, use a held output class to allow reviewing the messages later. Replace *message_class* with the appropriate message class value. Check your “Installation Conventions Worksheet” on page 2-14. for the appropriate value.

5 MSGLEVEL=(1,1) tells the system to reproduce this JCL code in the output, and to include allocation messages.

6 CLASS=*n* indicates the system resource requirements for the job. Check your "Installation Conventions Worksheet" on page 2-14. for the appropriate value.

7 The EXEC statement invokes the program IEFBR14 and identifies the first (and only) job step in this job. You are arbitrarily naming it STEP1. All of the control statements that follow the EXEC statement (until the next EXEC statement, if any) are part of this job step.

IEFBR14 is the name of a program within your MVS system. It does not actually process any data, but it enables you to run this job as a test to verify the JCL statements, and to create the input data. Later in the exercise you will replace IEFBR14 with the name of another program that sorts data.

8 SORTIN is the name you have given the DD statement that describes the input data.

9 NEPTUNE through JUPITER are the items to be sorted. This method of providing data to the program is referred to as *in-stream* data, an alternative to providing the input in a separate allocated data set.

10 /* indicates the end of the input data stream.

11 SORTOUT is the name you have given the DD statement that describes where the output from running the job will be placed. In this example, SYSOUT=* specifies that the output data will be directed to the SYSOUT device defined in the MSGCLASS statement.

12 /* (optional) denotes the end of the job.

For detailed information on each of the JCL statements and syntax requirements, refer to *OS/390 MVS JCL Reference*.

Step 3. Submit the JCL to the System as a Job

When you have finished entering the JCL into the data set, submit the job by entering the SUBMIT command from the ISPF EDIT command line, the TSO/E command line, or following a READY mode message. Each of these methods is shown below.

- ***ISPF EDIT command line:***

```
 EDIT ---- userid.SORT.JCL -------------------------- LINE 00000000 COL 001 080
 COMMAND ===> SUBMIT                                          SCROLL ===> CSR
******************************** TOP OF DATA  ********************************
//userid   JOB 'accounting data',
                      .
                      .
                      .
```

- ***TSO/E command line:***

```
------------------------  TSO COMMAND PROCESSOR  ---------------------------
ENTER TSO COMMAND OR CLIST BELOW:

===> SUBMIT 'userid.SORT.JCL'



ENTER SESSION MANAGER MODE ===> NO     (YES or NO)
```

- ***After READY mode message:***

```
    .
    .
    .
  READY
 SUBMIT 'userid.SORT.JCL'
```

Note: When entering the command from the TSO command line or after a READY message, you must surround the data set name with single quotation marks if you include your user ID. However, you can also enter the command without specifying your user ID and without using single quotation marks, as shown below:

```
SUBMIT SORT.JCL
```

When you do not specify the user ID and do not include single quotes, the system automatically inserts your user ID before the data set name. (The insertion of the user ID is for the duration of the current job; it is not a permanent change to the data set name.)

After entering the command, you should receive the following message indicating that your job was submitted successfully:

- ***When submitted from the ISPF EDIT command line:***

```
 EDIT ---- userid.SORT.JCL -------------------------- LINE 00000000 COL 001 080
 COMMAND ===> SUBMIT                                        SCROLL ===> CSR
******************************** TOP OF DATA  ********************************
//userid   JOB 'accounting data',
                      .
                      .
                      .
JOB jobname(jobnumber) SUBMITTED
***
```

- ***When submitted from the TSO command line:***

```
-------------------------  TSO COMMAND PROCESSOR  -----------
ENTER TSO COMMAND OR CLIST BELOW:

===> SUBMIT 'userid.SORT.JCL'




ENTER SESSION MANAGER MODE ===> NO     (YES or NO)
JOB jobname(jobnumber) SUBMITTED
***
```

- ***When submitted after READY mode message:***

```
    .
    .
    .
   READY
  SUBMIT 'userid.SORT.JCL'
    .
    .
    .
JOB jobname(jobnumber) SUBMITTED
***
    .
    .
   READY
```

When the job ends, you will receive a message indicating one of three conditions: job successful, JCL error, or program abend. If the message indicates the error or abend condition, review Steps 2 and 3 of this exercise to make sure that you followed the instructions exactly, then re-submit the job.

If the job fails again, consult the appropriate manual as indicated below:

If the message begins with HASP, the job was failed by JES2. For more information, refer to *OS/390 JES2 Messages*
If the message begins with IAT, the job was failed by JES3. For more information, refer to *OS/390 JES3 Messages*.

Step 4. View and Understand the Output from the Job

Use your installation's viewing facility (for example, SDSF) to view the output and determine whether the job completed successfully. (If you do not know how to use SDSF to view the output, see “Using SDSF to View Held Output from a Job” on page 2-17.)

If the job is on hold in the held queue, consider printing it for a record of the job activity.

```
1                  J E S 2  J O B  L O G  --  S Y S T E M  A Q T S  --  N O D E  P L P S C
0                                                                                                                    ----|
 15.21.28 JOB17653  IRR010I  USERID userid   IS ASSIGNED TO THIS JOB.                                                     |
 15.21.28 JOB17653  ICH70001I userid   LAST ACCESS AT 15:21:28 ON WEDNESDAY, OCTOBER 13, 1996                          |
 15.21.28 JOB17653  $HASP373 SORT  STARTED - INIT    9 - CLASS 5 - SYS AQTS                                           |
 15.21.28 JOB17653  IEF403I SORT - STARTED - TIME=15.21.28                                                            |
 15.21.28 JOB17653  - ============================================================================================== |
 15.21.28 JOB17653  -                                  REGION        --- STEP TIMINGS ---                  ----PAGING COUNTS----     |--- 1
 15.21.28 JOB17653  - STEPNAME PROCSTEP PGMNAME     CC     USED      CPU TIME  ELAPSED TIME    EXCP     SERV  PAGE  SWAP   VIO SWAPS |
 15.21.28 JOB17653  - STEP1             IEFBR14     00       4K   00:00:00.01   00:00:00.03       1      211     0     0     0     0 |
 15.21.28 JOB17653  IEF404I SORT - ENDED - TIME=15.21.28                                                              |
 15.21.28 JOB17653  - ============================================================================================== |
 15.21.28 JOB17653  - NAME-user_name        TOTALS: CPU TIME=  00:00:00.01  ELAPSED TIME=  00:00:00.05 SERVICE UNITS=    21          |
 15.21.28 JOB17653  - ============================================================================================== |
 15.21.28 JOB17653  $HASP395 SORT  ENDED                                                                         ----|
0------ JES2 JOB STATISTICS ------                                                                                ----|
-  13 OCT 1996 JOB EXECUTION DATE                                                                                      |
-           20 CARDS READ                                                                                            |
-           45 SYSOUT PRINT RECORDS                                                                                  |--- 2
-            0 SYSOUT PUNCH RECORDS                                                                                  |
-            3 SYSOUT SPOOL KBYTES                                                                                   |
-         0.00 MINUTES EXECUTION TIME                                                                                ----|
         1 //SORT JOB '662282,D58,9211064,S=C',                          JOB17653                                    ----|
           //  'user_name',                                                                                            |
           //  NOTIFY=userid,                                                                                         |
           //  MSGCLASS=H,                                                  00280009                                  |
           //  MSGLEVEL=(1,1),                                              00430010                                  |--- 3
           //  CLASS=5                                                      00430010                                  |
         2 //STEP1    EXEC  PGM=IEFBR14                                                                               |
         3 //SORTIN  DD *                                                                                            |
         4 //SORTOUT  DD SYSOUT=*                                                                                    |
         5 //SYSIN     DD *              GENERATED STATEMENT                                                       ----|
 ICH70001I userid   LAST ACCESS AT 15:21:28 ON WEDNESDAY, OCTOBER 13, 1996                                       ----|
 IEF236I ALLOC. FOR SORT STEP1                                                                                        |
 IEF237I JES2 ALLOCATED TO DATAIN                                                                                     |
 IEF237I JES2 ALLOCATED TO SYSIN                                                                                      |
 IEF142I SORT STEP1 - STEP WAS EXECUTED - COND CODE 0000  5                                                           |
 IEF285I   userid.SORT.JOB17653.D0000101.?           SYSIN                                                            |--- 4
 IEF285I   userid.SORT.JOB17653.D0000103.?           SYSOUT                                                           |
 IEF285I   userid.SORT.JOB17653.D0000102.?           SYSIN                                                            |
 IEF373I STEP /STEP1   / START 1996286.1521                                                                           |
 IEF374I STEP /STEP1   / STOP  1996286.1521 CPU    0MIN 00.01SEC SRB    0MIN 00.00SEC VIRT     4K SYS   180K EXT       4K SYS    9424K  |
 IEF375I  JOB /SORT / START 1996286.1521                                                                              |
 IEF376I  JOB /SORT / STOP  1996286.1521 CPU    0MIN 00.01SEC SRB    0MIN 00.00SEC                                ----|
```

Figure 2-2. Output from Job Invoking IEFBR14 Program

Figure 2-2 contains an example of the held output for this exercise. Each part of this output is explained below:

1 is installation-specific and may differ on your system.
2 contains JES messages about the job.
3 contains the JCL listing that resulted from the job.
4 contains the system messages resulting from processing the job.
5 condition code 0000 tells you that the program ran successfully. You receive one condition code for each step in the job. If a condition code is non-zero, see the documentation for the specific program you invoked. For more information on IEFBR14, see “Using IEFBR14 Program for Testing” on page 10-17.

Step 5. Make Changes to Your JCL

When your job has run successfully, edit the data set containing the JCL and change or add control statements as indicated below:

```
//SORT JOB 'accounting_data',
//  'user_name',
//  NOTIFY=&SYSUID,
//  MSGCLASS=H,
//  MSGLEVEL=(1,1),
//  CLASS=5
//STEP1    EXEC  PGM=SORT  1
//SYSIN   DD *
   SORT     FIELDS=(1,75,CH,A)  2
/*
//SYSOUT   DD  SYSOUT=*  3
//SORTIN   DD  *
NEPTUNE
PLUTO
EARTH
VENUS
MERCURY
MARS
URANUS
SATURN
JUPITER
/*
//SORTOUT  DD SYSOUT=*
/*
```

1 Replace the program name with the name of your sort program. In this job, SORT will sort the input data identified by the SORTIN DD statement.

2 Add the SYSIN control statement. SYSIN specifies how you want the sort to be done. In this case, you are indicating that you want to sort the fields from column 1 to column 75 as characters in ascending sequence.

3 Add the SYSOUT control statement. SYSOUT specifies the data set to which SORT will write its messages. A SYSOUT data set is a system-handled output data set. This data set is placed temporarily on direct access storage. Later, the system prints it or sends it to a specified location.

When you have finished entering the JCL into the data set, submit the job as you did in “Step 3. Submit the JCL to the System as a Job” on page 2-6.

Step 6. View and Understand Your Final Output

View your output as you did in “Step 4. View and Understand the Output from the Job” on page 2-8.

Figure 2-3 on page 2-11 shows an example of the held output for the completed exercise. Each part of this output is explained below:

```
1                  J E S 2  J O B  L O G  --  S Y S T E M  A Q T S  --  N O D E  P L P S C
0                                                                                                                    ----|
 13.40.27 JOB06572  IRR010I  USERID 'userid' IS ASSIGNED TO THIS JOB.                                                     |
 13.40.27 JOB06572  ICH70001I 'userid' LAST ACCESS AT 13:39:20 ON MONDAY, NOVEMBER 15, 1996                              |
 13.40.27 JOB06572  $HASP373 SORT     STARTED - INIT    9 - CLASS 5 - SYS AQTS                                          |
 13.40.27 JOB06572  IEF403I SORT - STARTED - TIME=13.40.27                                                              |
 13.40.28 JOB06572  - ============================================================================================== |
 13.40.28 JOB06572  -                                  REGION        --- STEP TIMINGS ---                  ----PAGING COUNTS----    |--- 1
 13.40.28 JOB06572  - STEPNAME PROCSTEP PGMNAME     CC     USED      CPU TIME  ELAPSED TIME    EXCP     SERV  PAGE  SWAP   VIO SWAPS |
 13.40.28 JOB06572  - STEP1             SORT        00     576K   00:00:00.03   00:00:00.15      20     1614     0     0     0     0 |
 13.40.28 JOB06572  IEF404I SORT - ENDED - TIME=13.40.28                                                                |
 13.40.28 JOB06572  - ============================================================================================== |
 13.40.28 JOB06572  - NAME-'user name'          TOTALS: CPU TIME=  00:00:00.03  ELAPSED TIME=  00:00:00.16 SERVICE UNITS=   1614       |
 13.40.28 JOB06572  - ============================================================================================== |
 13.40.28 JOB06572  $HASP395 SORT     ENDED                                                                           ----|
0------ JES2 JOB STATISTICS ------                                                                                    ----|
-  15 NOV 1996 JOB EXECUTION DATE                                                                                        |
-           25 CARDS READ                                                                                              |
-           81 SYSOUT PRINT RECORDS                                                                                    |--- 2
-            0 SYSOUT PUNCH RECORDS                                                                                    |
-            4 SYSOUT SPOOL KBYTES                                                                                     |
-         0.00 MINUTES EXECUTION TIME                                                                                 ----|
          1 //SORT JOB 'accounting data',                                  JOB06572                                   ----|
            //  'userid',                                                                                              |
            //  NOTIFY=&SYSUID,                                                                                        |
            //  MSGCLASS=H,                                                                                            |
            //  MSGLEVEL=(1,1),                                                                                        |
            //  CLASS=5                                                                                                |--- 3
          2 //STEP1    EXEC  PGM=SORT                                                                                  |
          3 //SYSIN   DD *                                                                                            |
          4 //SYSOUT   DD  SYSOUT=*                                                                                   |
          5 //SORTIN   DD  *                                                                                          |
          6 //SORTOUT  DD SYSOUT=*                                                                                    |
            /*                                                                                                       ----|
 ICH70001I 'userid' LAST ACCESS AT 13:39:20 ON MONDAY, NOVEMBER 15, 1996                                             ----|
 IEF236I ALLOC. FOR SORT STEP1                                                                                         |
 IEF237I JES2 ALLOCATED TO SYSIN                                                                                       |
 IEF237I JES2 ALLOCATED TO SYSOUT                                                                                      |
 IEF237I JES2 ALLOCATED TO SORTIN                                                                                      |
 IEF237I JES2 ALLOCATED TO SORTOUT                                                                                     |
 IEF142I SORT STEP1 - STEP WAS EXECUTED - COND CODE 0000 5                                                             |
 IEF285I   userid.SORT.JOB06572.D0000101.?              SYSIN                                                          |
 IEF285I   userid.SORT.JOB06572.D0000103.?              SYSOUT                                                         |
 IEF285I   userid.SORT.JOB06572.D0000102.?              SYSIN                                                          |
 IEF285I   userid.SORT.JOB06572.D0000104.?              SYSOUT                                                         |
 IEF373I STEP /STEP1   / START 1996319.1340                                                                            |
 IEF374I STEP /STEP1   / STOP  1996319.1340 CPU    0MIN 00.03SEC SRB    0MIN 00.00SEC VIRT   576K SYS   188K EXT    4096K SYS    9444K |
 IEF375I  JOB /SORT    / START 1996319.1340                                                                            |
 IEF376I  JOB /SORT    / STOP  1996319.1340 CPU    0MIN 00.03SEC SRB    0MIN 00.00SEC                                |
1ICE143I 0 BLOCKSET     SORT  TECHNIQUE SELECTED                                                                     |--- 4
 ICE000I 1 --- CONTROL STATEMENTS/MESSAGES ---- 5740-SM1 REL 12.0 ---- 13.40.28 NOV 15, 1996 --                        |
0             SORT     FIELDS=(1,75,CH,A)                                                                            |
 ICE088I 1 SORT    .STEP1    .       , INPUT LRECL = 80, BLKSIZE = 80, TYPE = F                                       |
 ICE093I 0 MAIN STORAGE = (MAX,4194304,4194304)                                                                        |
 ICE156I 0 MAIN STORAGE ABOVE 16MB = (3624960,3624960)                                                                 |
 ICE128I 0 OPTIONS: SIZE=4194304,MAXLIM=1048576,MINLIM=450560,EQUALS=N,LIST=Y,ERET=RC16 ,MSGDDN=SYSOUT                |
 ICE129I 0 OPTIONS: VIO=N,RESDNT=ALL ,SMF=NO   ,WRKSEC=Y,OUTSEC=Y,VERIFY=N,CHALT=N,DYNALOC=N            ,ABCODE=MSG   |
 ICE130I 0 OPTIONS: RESALL=4096,RESINV=0,SVC=109 ,CHECK=Y,WRKREL=Y,OUTREL=Y,CKPT=N,STIMER=Y,COBEXIT=COB1              |
 ICE131I 0 OPTIONS: TMAXLIM=4194304,ARESALL=0,ARESINV=0,OVERRGN=65536,EXCPVR=NONE ,CINV=Y,CFW=Y                       |
 ICE132I 0 OPTIONS: VLSHRT=N,ZDPRINT=N,IEXIT=N,TEXIT=N,LISTX=N,EFS=NONE    ,EXITCK=S,PARMDDN=DFSPARM ,FSZEST=N          |
 ICE133I 0 OPTIONS: HIPRMAX=OPTIMAL ,DSPSIZE=MAX                                                                       |
 ICE084I 0 BSAM ACCESS METHOD USED FOR SORTOUT                                                                         |
 ICE084I 0 BSAM ACCESS METHOD USED FOR SORTIN                                                                          |
 ICE090I 0 OUTPUT LRECL = 80, BLKSIZE = 80, TYPE = F                                                                   |
 ICE080I 0 IN MAIN STORAGE SORT                                                                                        |
 ICE055I 0 INSERT 0, DELETE 0                                                                                          |
 ICE054I 0 RECORDS - IN: 9, OUT: 9                                                                                     |
 ICE134I 0 NUMBER OF BYTES SORTED: 720                                                                                 |
 ICE180I 0 HIPERSPACE STORAGE USED = 0K BYTES                                                                          |
 ICE188I 0 DATA SPACE STORAGE USED = 0K BYTES                                                                          |
 ICE052I 0 END OF DFSORT                                                                                             ----|
 EARTH                                                                                                               ----|
 JUPITER                                                                                                               |
 MARS                                                                                                                  |
 MERCURY                                                                                                               |
 NEPTUNE                                                                                                               |--- 6
 PLUTO                                                                                                                 |
 SATURN                                                                                                                |
 URANUS                                                                                                                |
 VENUS                                                                                                              ---|
```

Figure 2-3. Output from Job Invoking SORT Program

1 is installation-specific and may differ on your system.
2 contains JES messages about the job.
3 contains the JCL listing that resulted from the job.
4 contains the system messages resulting from processing the job.
5 condition code 0000 tells you that the program ran successfully. You receive one condition code for each step in the job. If a condition code is

non-zero, see the documentation for the specific program you invoked (in this case, SORT).

6 contains the output produced by the SORT program.

More Complex Jobs

In-Stream and Cataloged Procedures

As you gain more experience in submitting jobs, you will find that you often use the same set of job control statements repeatedly with little or no change.

To save time and prevent errors, you can prepare sets of job control statements that you can execute again and again. You can do this through the use of two types of procedures: **in-stream procedures** and **cataloged procedures**.

In-Stream Procedures

An in-stream procedure is a named set of job control statements in a job that can be re-executed *within that job*, simply by invoking the name of the procedure. This enables you to execute the set of control statements more than one time in the same job without having to repeat the statements.

Figure 2-4 shows a job that contains an in-stream procedure. Its name is PTEST, and it ends with a PEND statement.

Figure 2-4. In-Stream Procedure

Job Control Statement	Explanation
`//JOB1 JOB CT1492,'JIM MOSER'`	Starts job
`//PTEST PROC`	Starts in-stream procedure
`//PSTA EXEC PGM=CALC`	Identifies first step in procedure
`//DDA DD DSNAME=D.E.F,DISP=OLD`	Request 3 data sets for first procedure step
`//DDB DD DSNAME=DATA1,DISP=(MOD,PASS)`	
`//DDOUT DD SYSOUT=*`	
`//PSTB EXEC PGM=PRNT`	Identifies second step in procedure
`//DDC DD DSNAME=*.PSTA.DDB,DISP=OLD`	Request 2 data sets for second procedure step
`//DDREP DD SYSOUT=A`	
`// PEND`	Ends in-stream procedure
`//STEP1 EXEC PROC=PTEST`	First step in JOB1, executes procedure
`//PSTA.IN DD *`	Adds in-stream data to procedure step PSTA
`.`	
`(data)`	
`.`	
`/*`	

Note: The maximum number of in-stream procedures you can code in any job is 15.

Cataloged Procedures

A cataloged procedure, like an in-stream procedure, is a named set of job control statements. However, these control statements are placed, or **cataloged**, in a partitioned data set (PDS) or partitioned data set extended (PDSE) known as a **procedure library**. This enables a cataloged procedure to be invoked *by any job*.

Cataloged procedures can be placed in the system procedure library SYS1.PROCLIB or in any user-specified procedure library (for example JCLLIB). For additional information on procedure libraries, refer to *OS/390 MVS JCL Reference*.

Figure 2-5 shows an example of a cataloged procedure named MYPROC. Figure 2-6 shows an example of a job that executes MYPROC.

Figure 2-5. Cataloged Procedure: Member MYPROC in SYS1.PROCLIB

Job Control Statement	Explanation
`//MYPROC PROC`	Starts cataloged procedure
`//MY1 EXEC PGM=WORK1`	Identifies first step in procedure
`//MYDDA DD SYSOUT=A`	Request 2 data sets for first procedure step
`//MYDDB DD SYSOUT=*`	
`//MY2 EXEC PGM=TEXT5`	Identifies second step in procedure
`//MYDDC DD DSNAME=F.G.H,DISP=OLD`	Request 2 data sets for second procedure step
`//MYDDE DD SYSOUT=*`	
`// PEND`	Indicate end of procedure.

Figure 2-6. Job that Executes Cataloged Procedure MYPROC

Job Control Statement	Explanation
`//JOB2 JOB ,'JACKIE DIGIAN'`	Starts job
`//STEPA EXEC PROC=MYPROC`	First step in JOB2, executes procedure
`//MY2.MYDDC DD DISP=(OLD,DELETE)`	Modifies DD statement MYDDC in procedure step MY2

Note: Before cataloging any procedure, test it as an instream procedure first.

Input Streams

Just as you can group several steps into one job, you can group several jobs together into one **input stream**. Any time jobs are placed in a series and entered through one input device, the series is considered an input stream. The input device can be a terminal, a magnetic tape device, or a direct access device.

Figure 2-7 shows a data set containing an input stream of three jobs.

Figure 2-7. Job Boundaries in a Three-Job Input Stream

	Job Control Statement	Explanation
Job 1	`//JOB1 JOB AT45,'GARY PUCHKOFF'`	First job
	`//STEP1 EXEC PGM=A33`	
	`//DDA DD DSNAME=CATDS,DISP=OLD`	
	`//DDB DD SYSOUT=A`	
Job 2	`//JOB2 JOB AT87,'JAN BUSKIRK'`	Second job
	`//STEPA EXEC PGM=REP`	
	`//DD1 DD *`	
	`.`	
	`(data)`	
	`.`	
	`//DD2 DD SYSOUT=C`	
Job 3	`//JOB3 JOB 1726,'MARK LAMAN'`	Third job
	`//ST1 EXEC PGM=ADDER`	
	`//DDIN DD DATA`	
	`.`	
	`(data)`	
	`.`	
	`/*`	
	`//DDOUT DD SYSOUT=A`	

Additional Information

Installation Conventions Worksheet

Using this worksheet, identify the conventions used at your MVS installation. Documenting this information will help you create JCL data sets that your system will accept. You may need to ask someone more familiar with your installation to help you identify the conventions indicated in the worksheet.

Convention		Installation-Specific Attribute(s)
Job Entry Subsystem (JES2/JES3)		
Data Set Editor		
Security Requirements		
Data Set Allocation	Volume Serial	
	Generic Unit	
	Space Units	
	Primary Quantity	
	Secondary Quantity	
	Directory Blocks	
	Record Format	`Fixed Block (RECFM=FB)`
	Logical Record Length	`80 (LRECL=80)`
	Block Size	• `0 for Sequential Data Sets` • `>0 for Partitioned Data Sets`
Job Information and Requirements	Accounting Data	
	Message Class	
	Input Processing Information	
	Output Processing Information	

Using ISPF to Allocate and Edit a Data Set

The following instructions explain how to use ISPF to allocate a data set, edit it, and place your JCL control statements in it.

Note: ISPF screens may differ slightly from one MVS installation to another.

1. On the ISPF Primary Option menu, select the appropriate item to display the Data Set Utility menu.
2. On the Data Set Utility menu, select Option A (allocate new data set) and enter a data set name as shown in step 3 below, replacing *userid* with your own user ID.

```
---------------------------  DATA SET UTILITY  ----------------------------
OPTION  ==   A

   A - Allocate new data set                C - Catalog data set
   R - Rename entire data set               U - Uncatalog data set
   D - Delete entire data set               S - Data set information (short)
   blank - Data set information             M - Enhanced data set allocation

ISPF LIBRARY:
   PROJECT ===>
   GROUP   ===>
   TYPE    ===>

OTHER PARTITIONED OR SEQUENTIAL DATA SET:
   DATA SET NAME  ===> 'userid.SORT.JCL'
   VOLUME SERIAL  ===>          (If not cataloged, required for option "C)

DATA SET PASSWORD ===>          (If password protected)
```

3. On the Allocate New Data Set menu, fill in the fields indicated in the example below, replacing *volser*, *unit*, and *size* with appropriate values according to the information you filled in on “Installation Conventions Worksheet” on page 2-14.

```
----------------------  ALLOCATE NEW DATA SET  --------------------------------
COMMAND ===>

DATA SET NAME: userid.SORT.JCL

   VOLUME SERIAL      ===> volser      (Blank for authorized default volume) *
   GENERIC UNIT       ===>             (Generic group name or unit address) *
   SPACE UNITS        ===> unit        (BLKS, TRKS, or CYLS)
   PRIMARY QUANTITY   ===> 1           (In above units)
   SECONDARY QUANTITY ===> 1           (In above units)
   DIRECTORY BLOCKS   ===> 0           (Zero for sequential data set)
   RECORD FORMAT      ===> FB
   RECORD LENGTH      ===> 80
   BLOCK SIZE         ===> size
   EXPIRATION DATE    ===>             (YY/MM/DD, YYYY/MM/DD
                                        YY.DDD, YYYY.DDD in Julian form
                                        DDDD for retention period in days
                                        or blank)

   ( * Only one of these fields may be specified)
```

4. Note that message “DATA SET ALLOCATED” indicates that the allocation has been completed.

```
 ---------------------------  DATA SET UTILITY  ----------- DATA SET ALLOCATED
```

5. Use ISPF to edit the allocated data set and enter the JCL control statements into the data set.

```
EDIT ---- userid.SORT.JCL ---------------------------------- COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> CSR
****** ***************************** TOP OF DATA ******************************
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
' ' ' ' ' ' '
****** **************************** BOTTOM OF DATA ****************************
```

6. If you are currently working on the exercise for creating and entering a JCL job, return to “Step 2. Edit the JCL Data Set and Add the Necessary JCL” on page 2-5 now.

Using SDSF to View Held Output from a Job

The following instructions explain how to use SDSF with a JES2 system to view the output from your job.

Note: SDSF screens may differ slightly from one JES2 installation to another. If you are using JES3, you can use (E)JES or a comparable tool to view the output.

1. ***Display the SDSF Primary Option Menu*** and select Option H

```
V1R4M0 NZ06 -------------  SDSF PRIMARY OPTION MENU  -------------------------
COMMAND INPUT ===> H                                         SCROLL ===> PAGE

 Type an option or command and press Enter.

    DA        - Display active users of the system
    I         - Display jobs in the JES2 input queue
    O         - Display jobs in the JES2 output queue
    H         - Display jobs in the JES2 held output queue
    ST        - Display status of jobs in the JES2 queues

    TUTOR     - Short course on SDSF (ISPF only)
    END       - Exit SDSF

    Licensed Materials - Property of IBM

    5665-488 (C) Copyright IBM Corp. 1981, 1993. All rights reserved.
    US Government Users Restricted Rights - Use, duplication or
    disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

2. ***To view an individual data set:***

   a. On the SDSF Held Output Display All Classes panel, enter a question mark (?) next to the job whose output data sets you want to view.

```
SDSF HELD OUTPUT DISPLAY ALL CLASSES   174       LINES LINE 1-2 (2)
COMMAND INPUT ===>                                    SCROLL ===> PAGE
PREFIX=*  DEST=(ALL)  OWNER=userid
NP  JOBNAME  JOBID    OWNER    PRTY C ODISP DEST                TOT-REC  TOT-P
?   jobname  JOB20482 userid      7 H HOLD  LOCAL
87
    jobname  JOB20517 userid      7 H HOLD  LOCAL                    87
               .
               .
               .
```

b. On the SDSF Job Data Set Display panel, enter the letter S next to the name of the data set you want to display.

```
SDSF JOB DATA SET DISPLAY - JOB useridS  (JOB20482)    LINE 1-5 (5)
COMMAND INPUT ===>                                          SCROLL ===>
PREFIX=*  DEST=(ALL)  OWNER=userid
NP  DDNAME   STEPNAME PROCSTEP DSID OWNER    C DEST              REC-CNT
S   JESMSGLG JES2                2 userid   H LOCAL                  22
    JESJCL   JES2                3 userid   H LOCAL                   6
    JESYSMSG JES2                4 userid   H LOCAL                  28
    SYSOUT   SORT              103 userid   H LOCAL                  22
    SORTOUT  SORT              104 userid   H LOCAL                   9
```

Note: On the above panel:

- JESMSGLG contains system messages.
- JESJCL contains JCL with procedures expanded, overrides applied, and symbolics resolved.
- JESYSMSG contains MVS system messages.
- SYSOUT contains messages produced by the program (in this case, SORT) executed in this job.
- SORTOUT contains the output produced by the program (in this case, SORT) executed in this job.

c. The system displays the selected data set (in this case, JESMSGLG):

```
1                  J E S 2  J O B  L O G  --  S Y S T E M  A Q T S  --  N O D E  P L P S C
0
 15.21.28 JOB17653  IRR010I  USERID userid   IS ASSIGNED TO THIS JOB.
 15.21.28 JOB17653  ICH70001I userid   LAST ACCESS AT 15:21:28 ON WEDNESDAY, OCTOBER 13, 1993
 15.21.28 JOB17653  $HASP373 SORT  STARTED - INIT    9 - CLASS 5 - SYS AQTS
 15.21.28 JOB17653  IEF403I SORT - STARTED - TIME=15.21.28
 15.21.28 JOB17653  - ------------------------------------------------------------------------------------------------------------------
 15.21.28 JOB17653  -                                  REGION        --- STEP TIMINGS ---                  ----PAGING COUNTS----
 15.21.28 JOB17653  - STEPNAME PROCSTEP PGMNAME     CC     USED      CPU TIME  ELAPSED TIME    EXCP      SERV  PAGE  SWAP   VIO SWAPS
 15.21.28 JOB17653  - STEP1             IEFBR14     00       4K   00:00:00.01   00:00:00.03       1      211     0     0     0     0
 15.21.28 JOB17653  IEF404I SORT - ENDED - TIME=15.21.28
 15.21.28 JOB17653  - ------------------------------------------------------------------------------------------------------------------
 15.21.28 JOB17653  - NAME-user_name        TOTALS: CPU TIME=  00:00:00.01  ELAPSED TIME=  00:00:00.05 SERVICE UNITS=
211
 15.21.28 JOB17653  - ------------------------------------------------------------------------------------------------------------------
 15.21.28 JOB17653  $HASP395 SORT  ENDED
```

3. ***To view the entire output:***

a. On the SDSF Held Output Display All Classes panel, enter the letter S next to the job whose output you want to see.

```
SDSF HELD OUTPUT DISPLAY ALL CLASSES   174       LINES LINE 1-2 (2)
COMMAND INPUT ===>                                    SCROLL ===> PAGE
PREFIX=*  DEST=(ALL)  OWNER=userid
NP  JOBNAME  JOBID    OWNER    PRTY C ODISP DEST                TOT-REC  TOT-P
S   jobname  JOB20482 userid      7 H HOLD  LOCAL                          87
    jobname  JOB20517 userid      7 H HOLD  LOCAL                    87
               .
               .
               .
```

b. You will be presented with one view of the entire output (as shown in Figure 2-3 on page 2-11).

Helpful Utilities

Figure 2-8 lists some common tasks that manage data sets, as well as utilities IBM provides that you can use to perform the tasks. For additional information on these utilities, see:

- *ISPF/PDF Guide and Reference*
- *DFSMS/MVS Access Method Services for ICF*
- *DFSMS/MVS Access Method Services for VSAM*
- *DFSMS/MVS Utilities*
- *OS/390 TSO/E User's Guide*

Other utility programs may be available to perform these and other system tasks.

Figure 2-8. Tasks and Utility Programs

Task	Utility Name
Allocate data sets	• TSO/E ALLOCATE command • ISPF/PDF Data Set Utility • Access Method Services ALLOCATE command • JCL DD statement, DISP=NEW parameter
Delete data sets	• TSO/E DELETE command • ISPF/PDF Data Set Utility • Access Method Services DELETE command • JCL DD statement, DISP=OLD,DELETE parameter
Compare data sets	IEBCOMPR (DFSMSdfp)
Copy data sets	IEBCOPY (DFSMSdfp)
Delete records in data sets	IEBUPDTE (DFSMSdfp)
Edit/print/punch data sets	IEBPTPCH (DFSMSdfp)
Insert records into data sets	IEBUPDTE (DFSMSdfp)
Merge data sets	IEBCOPY (DFSMSdfp)
Modify data sets	IEBUPDTE (DFSMSdfp)
Print data sets	IEBPTPCH (DFSMSdfp)
Rename members/data sets	IEBCOPY (DFSMSdfp)
Scratch data sets	IEHPROGM (DFSMSdfp)

Chapter 3. Job Control Tasks

For your program to execute on the computer and perform the work you designed it to do, your program must be processed by your operating system.

Your operating system consists of an MVS/SP base control program (BCP) with a job entry subsystem (JES2 or JES3) and DFSMS/MVS DFSMSdfp installed with it.

For the operating system to process a program, programmers must perform certain job control tasks. These tasks are performed through the job control statements, which consist of:

JCL statements
JES2 control statements
JES3 control statements

Entering Jobs

Job Steps

You enter a program into the operating system as a **job step**. A job step consists of the job control statements that request and control execution of a program and request the resources needed to run the program. A job step is identified by an EXEC statement. The job step can also contain data needed by the program. The operating system distinguishes job control statements from data by the contents of the records.

Jobs

A **job** is a collection of related job steps. A job is identified by a JOB statement.

Input Streams

Jobs placed in a series and entered through one input device form an **input stream**. The operating system reads an input stream into the computer from an input/output (I/O) device or an internal reader. The input device can be a card reader, a magnetic tape device, a terminal, or a direct access device. An internal reader is a buffer that is read from a program into the system as though it were an input stream.

Cataloged and In-Stream Procedures

You often use the same set of job control statements repeatedly with little or no change, for example, to compile, assemble, link-edit, and execute a program. To save time and prevent errors, you can prepare sets of job control statements and place, or catalog, them in a partitioned data set (PDS) or partitioned data set extended (PDSE) known as a procedure library. The data set attributes of a procedure library should match SYS1.PROCLIB (record length of 80 and record format of FB). Such a set of job control statements in the system procedure library, SYS1.PROCLIB (or an installation-defined procedure library), is called a **cataloged procedure**.

© Copyright IBM Corp. 1988, 1999

To test a procedure before placing it in the catalog, place it in an input stream and execute it; a procedure in an input stream is called an **in-stream procedure**. The maximum number of in-stream procedures you can code in any job is 15.

Steps in a Job

A job can be simple or complex; it can consist of one step or of many steps that call many in-stream and cataloged procedures. A job can consist of up to 255 job steps, including all steps in any procedures that the job calls. Specification of a greater number of steps produces a JCL error.

Processing Jobs

The operating system performs many job control tasks automatically. You can influence the way your job is processed by the JCL and JES2 or JES3 parameters you code. For example, the job entry subsystem selects jobs for execution, but you can speed up or delay selection of your job by the parameters you code.

Requesting Resources

Data Set Resources

To execute a program, you must request the data sets needed to supply data to the program and to receive output records from the program.

Sysout Data Set Resources

A sysout data set is a system-handled output data set. This data set is placed temporarily on direct access storage. Later, at the convenience of the system, the system prints it, punches it, or sends it to a specified location. Because sysout data sets are processed by the system, the programmer can specify many parameters to control that processing.

Task Charts

The following charts list the job control tasks, which are described in the *OS/390 MVS JCL User's Guide*, in four groups:

- Entering jobs in Figure 3-1 on page 3-3
- Processing jobs in Figure 3-2 on page 3-5
- Requesting data set resources in Figure 3-3 on page 3-6
- Requesting sysout data set resources in Figure 3-4 on page 3-8

For each task, the charts list the parameters and statements that can be used to perform it. In many cases, the same task can be performed using different parameters on different statements. Where a parameter can appear on both a JOB and EXEC statement, it applies to the entire job when coded on the JOB statement but only to a step when coded on an EXEC statement.

The system is designed to enable users to perform many types of job control in many ways. To allow this flexibility, only two job entry tasks are required:

- **Identification:** The job must be identified in the *jobname field* of a JOB statement.

- **Execution:** The program or procedure to be executed must be named in a PGM or PROC parameter on an EXEC statement.

Therefore, the following statements are the minimum needed to perform a job control task:

```
//jobname  JOB
//         EXEC  {PGM=program-name   }
                 {PROC=procedure-name}
                 {procedure-name}
```

Figure 3-1 (Page 1 of 3). Tasks for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Identification					
of job	jobname field		null statement (JES3 only)		
of step		stepname field			
of procedure			PROC PEND		
of INCLUDE group			INCLUDE		
of account	accounting information or pano in JOB JES2 accounting information	ACCT		/*NETACCT	//*NETACCT
of programmer	programmer's name and room in JOB JES2 accounting information USER			ROOM on /*JOBPARM	PNAME, BLDG, DEPT, ROOM, and USERID on //*NETACCT
Execution					
of program		PGM			
of procedure		PROC			
when restarting and with checkpointing	RESTART RD	RD	SYSCHK DD	RESTART on /*JOBPARM	FAILURE and JOURNAL on //*MAIN
deadline or periodic					DEADLINE on //*MAIN
when dependent on other jobs					//*NET
at remote node			XMIT JCL (JES3 only)	/*ROUTE XEQ /*XEQ /*XMIT	//*ROUTE XEQ
Job Input Control					
by holding job entrance	TYPRUN CLASS				HOLD, UPDATE, or CLASS on //*MAIN //*NET
by holding local input reader					//*PAUSE
by copying input stream (JES2 only)	TYPRUN CLASS				

Figure 3-1 (Page 2 of 3). Tasks for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
from remote work station				/*SIGNON /*SIGNOFF	/*SIGNON /*SIGNOFF
Communication					
from JCL to system			COMMAND Command	/*$command	//**command
from JCL to operator				/*MESSAGE	//*OPERATOR
from JCL to programmer	Comment field unless no parameter field	Comment field	//*comment, also comment field on all statements but null		Comment field on //*ENDPROCESS and //*PAUSE
from JCL to program		PARM			
from system to operator	WARNING on BYTES, CARDS, LINES, and PAGES				FETCH on //*MAIN WARNING on BYTES, CARDS, LINES, and PAGES on //*MAIN
from system to userid -of job completion -of print completion	NOTIFY		NOTIFY on OUTPUT JCL statement	/*NOTIFY	ACMAIN on //*MAIN with JOB NOTIFY
from TSO/E userid to system					USER on //*MAIN
from functional subsystem to programmer			PIMSG on OUTPUT JCL		
through job log	MSGCLASS MSGLEVEL log in JOB JES2 accounting information		JESDS on OUTPUT JCL	NOLOG on /*JOBPARM	
Protection					
through RACF	GROUP PASSWORD SECLABEL USER				
Resource Control					
of program library			JOBLIB DD, STEPLIB DD, DD defining PDS or PDSE member		

Figure 3-1 (Page 3 of 3). Tasks for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
of procedure library			JCLLIB	PROCLIB on /*JOBPARM	PROC and UPDATE on //*MAIN
of INCLUDE group			JCLLIB	PROCLIB on /*JOBPARM	PROC and UPDATE on //*MAIN
of address space	REGION ADDRSPC	REGION ADDRSPC			LREGION on //*MAIN
of processor				SYSAFF on /*JOBPARM	SYSTEM on //*MAIN
of spool partition					SPART and TRKGRPS on //*MAIN

Figure 3-2 (Page 1 of 2). Tasks for Processing Jobs

TASKS FOR PROCESSING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Processing Control					
by conditional execution	COND CANCEL on BYTES, CARDS, LINES, and PAGES	COND	IF/THEN/ELSE/ENDIF statement construct	CANCEL on BYTES, CARDS, LINES, and PAGES on /*JOBPARM	CANCEL on BYTES, CARDS, LINES, and PAGES on //*MAIN
by timing execution	TIME or time in JOB JES2 accounting information	TIME		TIME on /*JOBPARM	
for testing: 1. by altering usual processing 2. by dumping after error	TYPRUN CLASS DUMP on BYTES, CARDS, LINES, and PAGES	PGM=IEFBR14 PGM=JCLTEST PGM=JSTTEST (JES3 only)	SYSMDUMP DD SYSUDUMP DD SYSABEND DD To format dump on 3800 Printing Subsystem, FCB=STD3 and CHARS=DUMP on dump DD		//*PROCESS //*ENDPROCESS DUMP in BYTES, CARDS, LINES, and PAGES on //*MAIN
Performance Control					
by job class assignment	CLASS				CLASS on //*MAIN
by selection priority	PRTY			/*PRIORITY	
by performance group assignment	PERFORM	PERFORM			

Figure 3-2 (Page 2 of 2). Tasks for Processing Jobs

TASKS FOR PROCESSING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
by I/O-to-processing ratio					IORATE on //*MAIN

Figure 3-3 (Page 1 of 3). Tasks for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Identification					
of data set	DSNAME				UPDATE on //*MAIN
of in-stream data set	* or DATA SYSIN DD DLM		 /* or xx delimiter		//*DATASET //*ENDDATASET
of data set on 3540 Diskette Input/Output Unit	DSID				
through catalog	JOBCAT DD STEPCAT DD				
through label	label-type on LABEL				
by location on tape	data-set-sequence-number on LABEL				
as TCAM message data set	QNAME				
from or to terminal	TERM				
Description					
of status	DISP				
of data attributes - by modeling	DCB AMP DATACLAS KEYLEN DSNTYPE KEYOFF LRECL RECFM RECORG LIKE REFDD				
of data for ISO/ANSI Version 4 tapes	CCSID				

Figure 3-3 (Page 2 of 3). Tasks for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
of migration and backup	MGMTCLAS				
Protection					
through RACF	PROTECT SECMODEL				
for ISO/ANSI/FIPS Version 3 tapes and ISO/ANSI Version 4 tapes	ACCODE				
by passwords	PASSWORD and NOPWREAD on LABEL				
of access to BSAM and BDAM data sets	IN and OUT on LABEL				
Allocation					
of device	UNIT STORCLAS		CLASS on JOB (JES3 only)		SETUP and CLASS on //*MAIN
of tape or direct access volume	VOLUME STORCLAS				EXPDTCHK and RINGCHK on //*MAIN
of direct access space	SPACE AVGREC DATACLAS				
of virtual I/O	UNIT DSNAME= temporary data set				
with deferred volume mounting	DEFER on UNIT				
with volume pre-mounting				/*SETUP	
dynamic			DYNAMNBR on EXEC		
Processing Control					
by suppressing processing	DUMMY NULLFILE on DSNAME				
by postponing specification	DDNAME				
with checkpointing	CHKPT SYSCKEOV DD SYSCHK DD		RESTART on JOB RD on EXEC		
by subsystem	SUBSYS CNTL		CNTL ENDCNTL		
by TCAM job or task	QNAME				
End Processing					

Figure 3-3 (Page 3 of 3). Tasks for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
unallocation	FREE				
disposition of data set	DISP RETPD EXPDT			OUTDISP on /*OUTPUT	
release of unused direct access space	RLSE on SPACE				
disposition of volume	RETAIN and PRIVATE on VOLUME				

Figure 3-4 (Page 1 of 3). Tasks for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Identification					
as a sysout data set	SYSOUT				
name (last qualifier)	DSNAME				
of output class	class on SYSOUT	CLASS	MSGCLASS on JOB with SYSOUT=* or CLASS=* and SYSOUT=(,)		
of data set on 3540 Diskette Input/Output Unit	DSID				
Description					
of data attributes	DCB				
Protection					
of printed output		DPAGELBL SYSAREA			
Performance Control					
by queue selection		PRTY			
Processing Control					
with additional parameters	OUTPUT code-name on SYSOUT	DEFAULT			
by segmenting	SEGMENT				
with other data sets	class on SYSOUT	THRESHLD (JES3 only) GROUPID (JES2 only)			

Figure 3-4 (Page 2 of 3). Tasks for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
by external writer	writer-name on SYSOUT	WRITER			
by mode		PRMODE			
by holding	HOLD class on SYSOUT	CLASS OUTDISP			
by suppressing output	DUMMY class on SYSOUT	OUTDISP=PURGE on OUTPUT			
with checkpointing		CKPTLINE CKPTPAGE CKPTSEC		CKPLNS and CKPPGS on /*OUTPUT	
by Print Services Facility (PSF)		COLORMAP COMSETUP DUPLEX FORMDEF FORMLEN INTRAY OFFSETXB OFFSETXF OFFSETYB OFFSETYF OVERLAYB OVERLAYF PAGEDEF PRTERROR RESFMT USERLIB			
by IP Printway		PORTNO			
End Processing					
unallocation	FREE SPIN				
Destination Control					
to local or remote device or to another node	DEST class on SYSOUT	DEST COMPACT		/*ROUTE PRINT /*ROUTE PUNCH	ORG on //*MAIN
to another processor					ACMAIN on //*MAIN
to internal reader	INTRDR as writer-name on SYSOUT		/*EOF /*DEL /*PURGE /*SCAN		
to terminal	TERM				
to assist in sysout distribution		ADDRESS BUILDING DEPT NAME ROOM TITLE		ROOM on /*OUTPUT	
Output Formatting					

Figure 3-4 (Page 3 of 3). Tasks for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
to any printer	COPIES FCB form-name on SYSOUT UCS	COPIES FCB FORMS LINECT (JES2 only) UCS CONTROL	forms, copies, and linect on JOB JES2 accounting information	COPIES, FORMS, and LINECT on /*JOBPARM COPIES, FCB, and FORMS on /*OUTPUT	COPIES and FORMS on //*FORMAT PR
to 3800 Printing Subsystem in addition to most of printer parameters	BURST CHARS FLASH MODIFY DCB= OPTCD=J	BURST CHARS FLASH MODIFY TRC		BURST on /*JOBPARM CHARS, FLASH, and BURST on /*OUTPUT	CHARS and FLASH on //*FORMAT PR
to 3211 Printer with indexing feature		INDEX (JES2 LINDEX only)			
to punch	COPIES FCB form-name on SYSOUT DCB=FUNC=I	COPIES FCB FORMS			
of dumps on 3800 Printing Subsystem	CHARS=DUMP FCB=STD3	CHARS=DUMP FCB=STD3			
Output Limiting					
	OUTLIM		lines and cards on JOB JES2 accounting information BYTES, CARDS, LINES, and PAGES on JOB	BYTES, CARDS, LINES, and PAGES on /*JOBPARM	BYTES, CARDS, LINES, and PAGES on //*MAIN
USERDATA Specifications					
Installation specifications		USERDATA			

Part 2. Tasks for Entering Jobs

Chapter 4. Entering Jobs - Identification 4-1
Identification of Job 4-1
Identification of Step 4-2
Identification of Procedure 4-2
Identification of INCLUDE Group 4-3
Identification of Account 4-3
For Local Execution 4-3
For Remote Execution 4-4
Identification of Programmer 4-4

Chapter 5. Entering Jobs - Execution 5-1
Execution of Program 5-1
Execution of Procedure 5-2
Execution when Restarting and with Checkpointing (non-APPC) 5-2
Restarting after Abnormal Termination 5-2
Restarting When the System Failed in a JES2 System 5-3
Restarting When the System Failed in a JES3 System 5-3
Deadline or Periodic Execution in a JES3 System 5-4
Use of Deadline Scheduling 5-4
Use of Periodic Scheduling 5-4
Execution when Dependent on Other Jobs in a JES3 System 5-5
Execution at Remote Node (non-APPC) 5-7
Considerations when Submitting a Remote Job 5-8

Chapter 6. Entering Jobs - Job Input Control 6-1
Job Input Control by Holding Job Entrance (Non-APPC) 6-1
Job Input Control by Holding Local Input Reader (Non-APPC) 6-2
Job Input Control by Copying Input Stream (Non-APPC) 6-2
Job Input Control from Remote Work Station 6-3
JES2 Remote Job Entry 6-3
JES3 Remote Job Processing 6-4

Chapter 7. Entering Jobs - Communication 7-1
Communication from JCL to System (Non-APPC) 7-1
Communication from JCL to Operator (Non-APPC) 7-2
Communication from JCL to Programmer 7-2
Communication from JCL to Program 7-3
PARM Values for IBM-Supplied Programs 7-3
Communication from System to Operator 7-3
Messages during Volume Mounting 7-3
Messages When Job Exceeds Output Limit 7-4
Messages When Output Limit Exceeded in an APPC Scheduling Environment 7-4
Messages When Output Limit Exceeded in a Non-APPC Scheduling Environment 7-4
Use of Warning Messages 7-5
Communication from System to Userid 7-5
Job Completion 7-5
Print Completion 7-6
Communication from Time Sharing Userid to a JES3 System 7-6

© Copyright IBM Corp. 1988, 1999

Communication from Functional Subsystem to Programmer 7-6
Communication through Job Log 7-6
Printing Job Log and Sysout Data Sets Together 7-8

Chapter 8. Entering Jobs - Protection 8-1
Protection through RACF 8-1

Chapter 9. Entering Jobs - Resource Control 9-1
Resource Control of Program Library 9-1
System Library 9-1
Private Library 9-2
Temporary Library 9-4
Resource Control of Procedure Library 9-5
Retrieving a Procedure Library 9-5
Updating a Procedure Library 9-5
Resource Control of INCLUDE Group 9-6
Retrieving an INCLUDE Group 9-6
Resource Control of Address Space 9-6
Types of Storage 9-7
Requesting Amount and Type of Storage 9-7
Requesting Amount of Logical Storage in a JES3 System 9-9
Resource Control of the Processor 9-9
Selecting a Processor Using A Scheduling Environment 9-9
Selecting a Processor in JES2 9-10
Selecting a Processor in JES3 9-10
Resource Control of Spool Partitions in a JES3 System 9-11

This part describes how to enter jobs into the system. The tasks required to enter a job are:

- Identification
- Execution

Other tasks can optionally be performed:

- Job input control
- Communication
- Protection
- Resource control

Chapter 4. Entering Jobs - Identification

Figure 4-1. Identification Task for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Identification					
of job	jobname field		null statement (JES3 only)		
of step		stepname field			
of procedure			PROC PEND		
of INCLUDE group			INCLUDE		
of account	accounting information or pano in JOB JES2 accounting information	ACCT		/*NETACCT	//*NETACCT
of programmer	programmer's-name and room in JOB JES2 accounting information USER			ROOM on /*JOBPARM	PNAME, BLDG, DEPT, ROOM, and USERID on //*NETACCT

Identification of Job

Each job must be identified in the jobname field of the JOB statement. This identification is required and is coded:

```
//jobname JOB
```

The next JOB statement or the end of the input stream identifies the end of a job. A null statement can identify the end of a job or input stream.

Examples

```
//MYJOB    JOB
           .
           .
//MCS167   JOB
           .
           .
//R#123    JOB
           .
           .
//@5AB     JOB
           .
           .
//
```

© Copyright IBM Corp. 1988, 1999

This fifth statement is a null statement.

Identification of Step

A step name is required on only certain EXEC statements. In practice, name all steps. The system uses the step name in messages. If you omit the step name, the system leaves this field blank in messages, making it difficult to decide what step caused each message. A step name is coded:

```
//stepname EXEC
```

Examples

```
//STEP1    EXEC  PGM=A
              .
              .
//CHECK    EXEC  PROC=MHB15
              .
              .
//A$9      EXEC  PGM=RPTWRT
              .
              .
//MYPROGRM EXEC  PGM=CALC
              .
```

Identification of Procedure

For an in-stream procedure, identify the beginning with a PROC statement and the end with a PEND statement. Code a name on the PROC statement. The name for a TSO/E logon procedure should not be the same as the name of any subsystem.

For a cataloged procedure, PROC and PEND statements are optional. A PROC statement does not identify a cataloged procedure; the procedure is called by its member name or alias in the procedure library. However, use the PROC statement to assign default values for all symbolic parameters in the procedure. Then, if the calling EXEC statement or a SET statement does not assign a value to or nullify all the symbolic parameters, the step will not fail.

Examples

For in-stream procedures:

```
//PAYROLL PROC
             .
             .
//        PEND

//DESK3   PROC  A=NEWYORK,F=3350,C=(OLD,CATLG,DELETE)
             .
             .
//ENDING  PEND  THIS STATEMENT ENDS IN-STREAM PROCEDURE DESK3.
```

For cataloged procedures:

```
//      PROC  UT=3800,FM=J287,DT=LOCAL
```

Identification of INCLUDE Group

An INCLUDE statement identifies a member of a PDS or PDSE that contains a set of JCL statements. This set of JCL statements is called an INCLUDE group. The system replaces the INCLUDE statement with the statements in the INCLUDE group.

Example

The INCLUDE group INOUTDD contains:

```
//INOUT4   DD       DSNAME=DS4,UNIT=3380,VOL=SER=111112,
//                  DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
//INOUT5   DD       DSNAME=DS5,UNIT=3380,VOL=SER=111113,
//                  DISP=SHR
```

The system executes the following job step:

```
//STEP2    EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUT    INCLUDE  MEMBER=INOUTDD
//SYSUT3   DD       UNIT=SYSDS,SPACE=(TRK,(1))
//SYSUT4   DD       UNIT=SYSDS,SPACE=(TRK,(1))
COPYOPER   COPY     OUTDD=INOUT1
```

After the system executes the step, the JCL stream appears as follows:

```
//STEP2    EXEC     PGM=IEBCOPY
//SYSPRINT DD       SYSOUT=A
//INOUT4   DD       DSNAME=DS4,UNIT=3380,VOL=SER=111112,
//                  DISP=(NEW,KEEP),SPACE=(TRK,(5,1,2))
//INOUT5   DD       DSNAME=DS5,UNIT=3380,VOL=SER=111113,
//                  DISP=SHR
//SYSUT3   DD       UNIT=SYSDS,SPACE=(TRK,(1))
//SYSUT4   DD       UNIT=SYSDS,SPACE=(TRK,(1))
COPYOPER   COPY     OUTDD=INOUT1
```

Identification of Account

For Local Execution

In JES initialization parameters, the installation specifies whether or not accounting information is required in the accounting information parameter on the JOB statement and/or the ACCT parameter on the EXEC statement. The installation decides what accounting information is needed and the format for the information.

Examples

```
//J28  JOB (12A75,DEPTD58,921)
        .
        .
//XYZ  JOB '12A75,DEPTD58,921'

If a subparameter contains special characters:

//GHI  JOB (12A75,'DEPT/D58',921)
        .
        .
//JKL  JOB '12A75,DEPT/D58,921'

If only an account number is coded:

//MNO  JOB 12A75
        .
        .
//PQR  JOB '12A.75'

If the account number is omitted:

//STU  JOB (,DEPTD58,921)
```

For Remote Execution

The JES2 /*NETACCT statement and the JES3 //*NETACCT statement supply accounting information for jobs sent to remote nodes for execution.

Examples

```
For remote execution in a JES2 system:

/*NETACCT 27FD16

For remote execution in a JES3 system:

//*NETACCT PNAME=FKRUPA,ACCT=27FD16,BLDG=921,DEPT=D58,
        .
        .
//*NETACCT ROOM=2T13,USERID=DDFKPGMR
```

Identification of Programmer

In JES initialization parameters, the installation specifies if a programmer's-name parameter is required on the JOB statement. The installation decides what the parameter must contain.

Examples

```
//ABC  JOB  ,L.GORDON
        .
        .
//DEF  JOB  ,'L GORDON'
        .
        .
//GHI  JOB  ,'SP/4 L. GORDON'
        .
        .
//JKL  JOB  ,'DEPT. 7202'
```

The USER parameter can be coded on the JOB statement to identify the person submitting the job.

Example

```
//MNO  JOB  ACCT15,'DON PIZZUTO',USER=ID32DBP
```

Chapter 5. Entering Jobs - Execution

Figure 5-1. Execution Task for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Execution					
of program		PGM			
of procedure		PROC			
when restarting and with checkpointing	RESTART RD	RD	SYSCHK DD	RESTART on /*JOBPARM	FAILURE and JOURNAL on //*MAIN
deadline or periodic					DEADLINE on //*MAIN
when dependent on other jobs					//*NET
at remote node			XMIT JCL (JES3 only)	/*ROUTE XEQ /*XEQ /*XMIT	//*ROUTE XEQ

Execution of Program

All programs to be executed must reside in a library, which is a partitioned data set (PDS) or partitioned data set extended (PDSE). The installation should maintain a list of programs available in its libraries. Libraries are of three types:

- System libraries: such as SYS1.LINKLIB
- Private libraries: specified in a JOBLIB or STEPLIB DD statement
- Temporary libraries: created in a previous step of the job.

For information about libraries, see “Resource Control of Program Library” on page 9-1.

Execute a program in a system or private library by coding:

```
//stepname  EXEC  PGM=program-name
```

Execute a program in a temporary library by coding:

```
//stepname  EXEC  PGM=*.stepname.ddname
//stepname  EXEC  PGM=*.stepname.procstepname.ddname
```

Examples

```
//ST1    EXEC  PGM=MYPROG
//DSPROG DD    DSNAME=PDS1(MEMP),DISP=SHR
//ST2    EXEC  PGM=*.ST1.DSPROG
```

© Copyright IBM Corp. 1988, 1999

Execution of Procedure

A procedure to be executed must be a:

- In-stream procedure, located in the input stream before the EXEC statement that calls it.
- Cataloged procedure, defined in the system procedure library concatenation SYS1.PROCLIB, an installation-defined procedure library, or a private library.

Execute an in-stream or cataloged procedure by coding:

```
//stepname  EXEC  PROC=procedure-name
//stepname  EXEC  procedure-name
```

Examples

```
//ST1    EXEC  PROC=PROCA
//STEP9  EXEC  PROC=DAILY
```

Execution when Restarting and with Checkpointing (non-APPC)

In an APPC scheduling environment, job restart is not supported.

Restarting after Abnormal Termination

If a job terminates abnormally, the checkpoint/restart facilities allow you to restart the job, as follows:

- Automatic step restart, that is, restart by the system from the beginning of a job step.
- Automatic checkpoint restart, that is, restart by the system from a checkpoint within a job step.
- Deferred step restart, that is, restart at a later time from the beginning of a job step.
- Deferred checkpoint restart, that is, restart at a later time from a checkpoint within a job step.

Restarts are controlled by:

- RD parameters on JOB and EXEC statements. (Restart is not supported for started tasks; do not use the RD parameter on the JOB statement for a started task.)
- Checkpoints, if written. Each time a CHKPT macro is executed, a checkpoint is written.
- The job journal, which is only required for an automatic restart. In a JES3 system, the programmer can code a JOURNAL parameter on the JES3 //*MAIN statement to control whether JES3 creates a journal for the job.
- In deferred restarts, a RESTART parameter on the JOB statement for the restarting job and a SYSCHK DD statement to identify the data set containing the checkpoint written in response to the CHKPT macro. (Restart is not supported for started tasks; do not use the RESTART parameter on the JOB statement for a started task.)

Use of Restart

Either form of restart saves having to execute the job from its beginning. If the job is long, restarting can save a lot of time and computer resources.

For more information about restarting, see *DFSMS/MVS Checkpoint/Restart*.

Examples

```
//J1    JOB   ,'B. MORRISON',RD=RNC

//J2    JOB   ,'H. MORRILL'
//S1    EXEC  PGM=TESTING,RD=R
//S2    EXEC  PGM=TESTED,RD=NC
```

Restarting When the System Failed in a JES2 System

JES2 requeues the job for execution if RESTART=Y is in the JES2 /*JOBPARM statement, and all of the following conditions apply:

- The job was executing when the system failed
- The operator reinitializes the system with a JES2 warm start
- The job cannot restart from a step or a checkpoint.

Re-execution is from the beginning of the job.

If the job is registered with automatic restart management, automatic restart management overrides RESTART=N, and queues the job for re-execution.

For more information about using automatic restart management, see *OS/390 MVS Setting Up a Sysplex* and *OS/390 MVS Programming: Sysplex Services Guide*.

Example

```
//J3    JOB  ,'J. BUSKIRK'
/*JOBPARM  RESTART=Y
            .
            .
```

Restarting When the System Failed in a JES3 System

If the job was executing when the system failed, the FAILURE parameter on the JES3 //*MAIN statement tells JES3 how to handle the job. The job can be restarted, cancelled, held, or printed and then held for restart.

If the job is registered with automatic restart management, automatic restart management overrides the value of the FAILURE= keyword, and queues the job for re-execution.

For more information about using automatic restart management, see *OS/390 MVS Setting Up a Sysplex* and *OS/390 MVS Programming: Sysplex Services Guide*.

Example

```
//J4    JOB  ,'G. HILL',RD=NC
//*MAIN FAILURE=RESTART
            .
            .
```

Deadline or Periodic Execution in a JES3 System

Use the DEADLINE parameter on the JES3 //*MAIN statement to execute your job by a certain time or periodically every week, month, or year. As the deadline approaches, JES3 increases the job's priority until it is executed. The priority is increased according to the installation-defined algorithm requested in the second subparameter.

Note: The term 'periodically' means that you submit a job as many times as you need it to process. For example, if you need a job to run once a month for every month of the year, you would submit 12 jobs with a date for each month. You could not submit a job once and have it process 12 times.

Use of Deadline Scheduling

The purpose of deadline scheduling is to help JES3 use available resources best. For example, if you work first shift and submit a job at the end of the day, you do not need output until the next morning. Specify 7 a.m. of the next day in the DEADLINE parameter and assign the job a low priority. JES3 can schedule the job any time during the night when the resources are available. But, if the job has not been scheduled by several hours before 7 a.m., JES3 increases its priority. JES3 will increase the job's priority periodically until it is selected for execution by 7 a.m.

Examples

To execute a job by 7 a.m. on January 20, 1986, code:

```
//*MAIN DEADLINE=(0700,B,012086)
```

The syntax changes slightly if you specify a date on or after the year 2000.

To execute a job by 7 a.m. on January 20, 2000, code:

```
//*MAIN DEADLINE=(0700,B,01/20/2000)
```

Use of Periodic Scheduling

The purpose of periodic scheduling is to run certain weekly, monthly, or yearly programs automatically.

Examples

To execute a job by 2 p.m. every Friday, code:

```
//*MAIN DEADLINE=(1400,A,6,WEEKLY)
```

Execution when Dependent on Other Jobs in a JES3 System

Use dependent job control (DJC) when jobs must be executed in a specific order. The group of jobs that depend on each other form a **dependent job control (DJC) network**. To indicate to JES3 the relationship of jobs to each other in a DJC network, code a JES3 //*NET statement in each job. Jobs in a network are of two types:

- Predecessor jobs, which must be completed before another job.
- Successor jobs, which must not be executed until one or more jobs are completed.

Using parameters on the //*NET statement, you can make execution of a job depend on how a predecessor terminated: normally or abnormally. When a predecessor job completes, a successor job:

- Can have the count of predecessor jobs it is waiting for decreased by one. When the count reaches zero, the successor job is queued for execution.
- Can be flushed from the system. The successor job and all of its successors are canceled, printed, and flushed from the system.
- Can be retained until the operator releases it. The successor job and all of its successors are kept from being scheduled. The job is released only when its immediate predecessor is resubmitted or the operator decreases the predecessor job number.

External Dependencies

If your job depends on external events, you can specify a count of predecessor jobs that is one greater than needed. The system will hold the job because the count cannot reach zero. When the external event occurs, the operator can issue a *MODIFY,N command to reduce the number so that the job will execute.

Testing a Network

To test a network without executing the programs, substitute the following for each actual EXEC statement:

```
//stepname EXEC PGM=IEFBR14
```

Example 1

To set up a DJC network, first draw a diagram of the dependencies:

```
JOBA  JOBB
   |  |
   JOBC
   |  |
JOBD  JOBE
```

Give the network a name: XMP1. This is the //*NET statement NETID parameter.

Then list each job and its predecessors and successors:

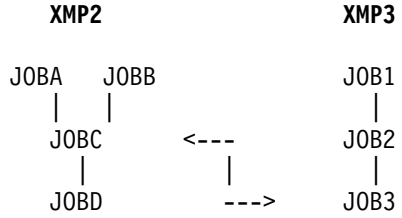
jobname	Predecessors //*NET NHOLD	Successors //*NET RELEASE
JOBA	0	JOBC
JOBB	0	JOBC
JOBC	2	JOBD, JOBE
JOBD	1	none
JOBE	1	none

Finally, code a //*NET statement to appear in each job:

```
//JOBA  JOB  ...
//*NET  NETID=XMP1,RELEASE=(JOBC)
//S1    EXEC ...
         .
         .
//JOBB  JOB ...
//*NET  NETID=XMP1,RELEASE=(JOBC)
//SA    EXEC ...
         .
         .
//JOBC  JOB ...
//*NET  NETID=XMP1,NHOLD=2,RELEASE=(JOBD,JOBE)
//S1    EXEC ...
         .
         .
//JOBD  JOB ...
//*NET  NETID=XMP1,NHOLD=1
//SA    EXEC ...
         .
         .
//JOBE  JOB ...
//*NET  NETID=XMP1,NHOLD=1
//S1    EXEC ...
         .
```

Example 2

This example shows two networks. JOB3 in network XMP3 depends on JOBC in network XMP2.



jobname	Predecessors //*NET NHOLD	Successors //*NET RELEASE
JOBA	0	JOBC
JOBB	0	JOBC
JOBC	2	JOB3
JOBD	1	none
JOB1	0	JOB2
JOB2	1	JOB3
JOB3	2	none

The //*NET statements for each job are:

```
For JOBA:   //*NET NETID=XMP2,RELEASE=(JOBC)
For JOBB:   //*NET NETID=XMP2,RELEASE=(JOBC)
For JOBC:   //*NET NETID=XMP2,NHOLD=2,NETREL=(XMP3,JOB3),RELEASE(JOBD)
For JOBD:   //*NET NETID=XMP2,NHOLD=1
For JOB1:   //*NET NETID=XMP3,RELEASE=(JOB2)
For JOB2:   //*NET NETID=XMP3,NHOLD=1,RELEASE=(JOB3)
For JOB3:   //*NET NETID=XMP3,NHOLD=2
```

Execution at Remote Node (non-APPC)

JES control statements and the XMIT statement have no function in an APPC scheduling environment.

You can enter a job through your system to execute on another system by coding one of the following statements. The job can be entered through an input reader, an internal reader, a TSO/E terminal, or an RJE (remote job entry) or RJP (remote job processing) terminal or work station.

When Entered through a JES2 System:

- And received by a JES2 system, code one of the following:

  ```
  /*ROUTE XEQ node
  /*XEQ node
  ```

- And received by a JES2 system or a JES3 system, code:

  ```
  /*XMIT node
  ```

- And received by a VM system with an MVS system running as a guest, code one of the following:

  ```
  /*ROUTE XEQ node.vmguestid
  /*XEQ node.vmguestid
  /*XMIT node.vmguestid
  ```

When Entered through a JES3 System:

- And received by a system other than a VM system, code:

```
//name  XMIT  DEST=node,DLM=xx
```

- And received by a VM system with another system running as a guest, code:

```
//name  XMIT  DEST=node.vmuserid,DLM=xx
```

Use of XMIT JCL Statement in a JES3 System

A //*ROUTE XEQ statement can also be used to transmit records from a JES3 node. Because an XMIT JCL statement allows transmission of records that the //*ROUTE XEQ statement does not allow, use XMIT JCL statements rather than //*ROUTE XEQ statements.

For example, a JOB statement for the receiving node must immediately follow a //*ROUTE XEQ statement. This requirement means that a //*ROUTE XEQ statement cannot be used to transmit records beginning with $$ POWER control statements to a VSE node; however, an XMIT JCL statement can transmit such records.

Considerations when Submitting a Remote Job

When submitting a job for remote execution, find out the installation-determined attributes of the executing system. Code these values in your JCL for the job.

The content and format of the JOB statement: Code the executing system's parameters on the JOB statements that the executing system will process.
The JES of the executing system: Code your JES control statements and JCL parameters for the executing system's JES.
The content of SYS1.PROCLIB in the executing system: Call only procedures available in the executing system.
The data sets at the executing system: Use only data sets that are available at the executing system, with the DD parameters that the executing system requires.
Installation-specific device names: Code only UNIT names used by the executing system.
The sysout classes at the executing system: Specify the executing system's sysout classes that have the attributes you need.
The job classes at the executing system: Specify the executing system's job class that has the attributes you need.

Examples

```
//MYJOB  JOB  27D15,'DON SMITH'
//TRANS  XMIT DEST=FARSYS
//THEIRJOB  JOB (DLD1,2E44),'POK LAB'
//*MAIN JOURNAL=YES
//S1    EXEC  PROC=RR23,A=3350,
//     C=25,DP=OLD
/*
```

- Job MYJOB is processed by the submitting JES3 location
- XMIT TRANS sends the following job to FARSYS
- THEIRJOB is sent as JOB statement; processed by FARSYS

Chapter 6. Entering Jobs - Job Input Control

Figure 6-1. Input Control Task for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Job Input Control					
by holding job entrance	TYPRUN CLASS				HOLD, UPDATE, or CLASS on //*MAIN //*NET
by holding local input reader					//*PAUSE
by copying input stream (JES2 only)	TYPRUN CLASS				
from remote work station				/*SIGNON /*SIGNOFF	/*SIGNON /*SIGNOFF

Job Input Control by Holding Job Entrance (Non-APPC)

The job input control described in this topic is not supported in an APPC scheduling environment. The TYPRUN and CLASS parameters are not supported for started tasks. If TYPRUN is specified, the job will fail. If CLASS is specified, it is ignored.

If a job must wait for an external event before it can execute, use one of the following. JES holds the job until the system operator releases it or until an event occurs.

In a JES2 system

- TYPRUN=HOLD or TYPRUN=JCLHOLD on the JOB statement. The operator must release the job.
- A JOB statement CLASS that requests a job class defined during JES2 initialization as held. The operator must release the job.

In a JES3 system

- TYPRUN=HOLD or CLASS on the JOB statement or HOLD=YES or CLASS on the //*MAIN statement. The operator must release the job.
- A job in a dependent job net; see “Execution when Dependent on Other Jobs in a JES3 System” on page 5-5. JES3 releases the job when the other job(s) complete execution, or the operator releases the job.
- UPDATE on the //*MAIN statement of another job, if this job would use the procedure library being updated or any library concatenated to it. JES3 releases the job when the updating job completes execution.

Use of Job Holding

You may need to delay execution of a job for several reasons. For example:

© Copyright IBM Corp. 1988, 1999

- If one job is updating a data set that another job must use.
- If the resources a job requires may not be available until an external event occurs.

Note: You cannot depend on job priorities to control the order in which jobs execute. The priority specified in the JOB statement PRTY parameter or in the JES2 /*PRIORITY statement affects the selection order. It does not guarantee that a job with a higher priority will complete execution before a job with a lower priority is started.

Examples

```
//J1   JOB   ,'J. COLE',TYPRUN=HOLD
       .
//J2   JOB   ACCT1734,'T. CURATOLO',CLASS=H

//*MAIN HOLD=YES
//*MAIN UPDATE=DS3
```

Job Input Control by Holding Local Input Reader (Non-APPC)

The //**PAUSE statement is not supported in an APPC scheduling environment. If you code //**PAUSE, the system will ignore it, and it will appear as a comment in the job listing.

In a JES3 system, use a //**PAUSE statement to halt an input reader. JES3 issues a message and waits for the operator to issue a *START command or for a remote work station with console level 15 to send a start message.

Example

```
//**PAUSE
//FIRST  JOB  ,'D. SCHOFER'
         .
         .
```

Job Input Control by Copying Input Stream (Non-APPC)

The job input control described in this section is not supported in an APPC scheduling environment.

In a JES2 system, code one of the following on the JOB statement to copy an input job, without executing any steps. While copying the input stream, JES2 scans the JCL for syntax errors.

- TYPRUN=COPY
- A CLASS job class defined during JES2 initialization as containing jobs to be copied without execution.

In both cases, JES2 places the copy of the input stream in a sysout data set. The sysout data set is in the class specified in the JOB statement MSGCLASS parameter. Pick the MSGCLASS class to control how the copied input stream is to be processed, as follows:

- By JES2 or by an external writer.

- Scheduled for immediate output or held because the message class is held. If held, the sysout data set is available to the TSO/E OUTPUT command.

Examples

```
//CPYJ1  JOB  1589D10,'I. BUTLER',TYPRUN=COPY
          .
//CPYJ2  JOB  ,'D. BALLARD',CLASS=P
          .
```

Job Input Control from Remote Work Station

JES2 Remote Job Entry

JES2 remote job entry (RJE) allows a remote work station to submit a job to a distant system and have the job processed by the system's JES2. Your installation's security product can control RJE stations. The output can be retained at the host system, sent to the work station, or sent to another location. JES2 processes a remote job as if it had been submitted locally. The remote station becomes a logical extension of the computer system that processes its jobs.

JES2 supports two ways of communicating with RJE remote stations:

- Through **systems network architecture synchronous data link control (SNA/SDLC)** protocol. SNA stations gain access to JES2 through VTAM.
- Through **binary synchronous communication (BSC)** protocol. Communication between the local processor and a BSC RJE station uses a JES2 facility called *multi-leaving*. Multi-leaving allows transmission of multiple print and punch streams at the same time and allows JES2 to receive multiple console messages and input streams.

For more information, see remote job entry in *OS/390 JES2 Initialization and Tuning Guide* and *OS/390 eNetwork Communications Server: SNA Programming*.

JES2 expects the remote station to be under the control of a remote operator. The RJE stations can consist of two types of devices:

- **Remote terminal**, which does not have a processor. A remote terminal, for example a 2780 or 2770, can be used to enter jobs into and receive data from JES2.
- **Remote work station**, which has a processor. A processor, for example a System/370 or System/390, executes a JES2-generated program that allows the processor to send jobs to and receive data from JES2. The remote work station also includes printers, punches, card readers, and a console.

Remote Job Entry Stations

During JES2 initialization, installations can configure remote lines as dedicated or nondedicated. For nondedicated remote lines, use the following to notify JES2 that you wish to begin and end a remote job stream processing session:

- For SNA remote work stations: the LOGON command to begin and either the LOGOFF command or the JES2 /*SIGNOFF control statement to end.

- For BSC remote work stations: the JES2 /*SIGNON control statement to begin and the JES2 /*SIGNOFF control statement to end.

For a discussion of the LOGON and LOGOFF commands, refer to *OS/390 JES2 Initialization and Tuning Reference* and *OS/390 eNetwork Communications Server: SNA Programming*.

JES3 Remote Job Processing

JES3 remote job processing (RJP) allows a remote work station to submit a job through a data link to a distant global processor and have the job processed by the system's JES3. The output can be retained at the host system, sent to the work station, or sent to another location. JES3 processes a remote job as if it had been submitted locally.

Devices attached to a processor by channels are **local devices**; devices attached to a processor by a data link are **remote devices**.

JES3 supports two ways of communicating with RJP remote devices:

- Through **systems network architecture synchronous data link control (SNA/SDLC)** protocol.
- Through **binary synchronous communications (BSC)** protocol.

Remote Work Stations

During JES3 initialization, installations can configure remote lines as dedicated or nondedicated. For nondedicated remote lines, use the following to notify JES3 that you wish to begin and end a remote job stream processing session:

- For SNA remote work stations: the LOGON command to begin and either the LOGOFF command or the JES3 /*SIGNOFF control statement to end.
- For BSC remote work stations: the JES3 /*SIGNON control statement to begin and the JES3 /*SIGNOFF control statement to end.

For a discussion of the LOGON and LOGOFF commands, refer to *OS/390 JES3 Initialization and Tuning Reference* and *OS/390 eNetwork Communications Server: SNA Programming*.

Chapter 7. Entering Jobs - Communication

Figure 7-1. Communication Task for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Communication					
from JCL to system			COMMAND Command	/*$command	//**command
from JCL to operator				/*MESSAGE	//*OPERATOR
from JCL to programmer	Comment field unless no parameter field	Comment field	//*comment, also comment field on all statements but null		Comment field on //*ENDPROCESS and //*PAUSE
from JCL to program			PARM		
from system to operator	WARNING on BYTES, CARDS, LINES, and PAGES				FETCH on //*MAIN WARNING on BYTES, CARDS, LINES, and PAGES on //*MAIN
from system to userid -of job completion -of print completion	NOTIFY		NOTIFY on OUTPUT JCL statement	/*NOTIFY	ACMAIN on //*MAIN with JOB NOTIFY
from TSO/E userid to system					USER on //*MAIN
from functional subsystem to programmer			PIMSG on OUTPUT JCL		
through job log	MSGCLASS MSGLEVEL log in JOB JES2 accounting information		JESDS on OUTPUT JCL	NOLOG on /*JOBPARM	

Communication from JCL to System (Non-APPC)

The statements described in this section are not supported in an APPC scheduling environment.

Use the following to communicate from your JCL to the system:

© Copyright IBM Corp. 1988, 1999

- In a JES2 system,
  - The JCL COMMAND statement to enter any MVS and JES commands that can be issued from the operator's console
  - The JCL command statement to enter system operator commands
  - The JES2 /*$command statement to enter JES2 commands.
- In a JES3 system,
  - The JCL COMMAND statement to enter any MVS and JES commands that can be issued from the operator's console
  - The JCL command statement to enter system operator commands
  - The JES3 //**command statement to enter JES3 commands.

The system executes any in-stream command as soon as it is read. Therefore, the command will not be synchronized with the execution of any job or step.

Examples

```
In a JES2 system:
     /*$SI3-5

     //    COMMAND  'CANCEL MYJOB,DUMP'

In a JES3 system:
     //**START
```

Communication from JCL to Operator (Non-APPC)

Use a /*MESSAGE control statement in a JES2 system or a //*OPERATOR control statement in a JES3 system to send a message to the operator when JES reads the job from the input stream. Note that the message is not synchronized with the execution of any job or step.

Examples

```
In a JES2 system:
     /*MESSAGE JOB J67 IS HELD. CALL X65335 BEFORE RELEASING J67.

In a JES3 system:
     //*OPERATOR JOB J67 IS HELD. CALL X65335 BEFORE RELEASING J67.
```

Communication from JCL to Programmer

To communicate from your JCL to programmers, use comments fields or JCL //*comment statements. The comments appear in the job log output listing if the JOB statement MSGLEVEL parameter requests that the statements be printed.

Use comments primarily to document your job and its resource requirements.

Examples

```
//* JOB J67 IS HELD UNTIL THE OPERATOR RELEASES IT.
//* THE OPERATOR SHOULD RELEASE J67 WHEN DISK 398
//* IS AVAILABLE.
```

Communication from JCL to Program

A processing program can require information that can vary from execution to execution. For example, the assembler and the linkage editor require that the programmer supply options and module attributes at execution. To provide information to a program, code the PARM parameter on the EXEC statement that executes the program.

To use the information, the processing program must contain instructions to retrieve the information. Retrieval of the PARM information is detailed in *OS/390 MVS Programming: Assembler Services Guide*.

Examples

```
//FIRST EXEC  PGM=IEV90,PARM=(OBJECT,NODECK,'LINECOUNT=50')
//LATER EXEC  PGM=HEWL,PARM='XREF,LIST,LET'
```

PARM Values for IBM-Supplied Programs

Some IBM-supplied programs allow you to select options from a set of alternatives. The PARM values are listed in the publication for the program. For many IBM-supplied programs, default values can be assigned to PARM values during system initialization. That is, the installation can select an alternative or assign a fixed value. The system uses this default unless you specify another value in the PARM parameter when you execute the IBM-supplied program.

The installation should maintain a list of default values assigned during system initialization.

Communication from System to Operator

The system sends to the operator console messages deemed to be needed by the operator.

Messages during Volume Mounting

In a JES3 system, the programmer can control the fetch messages that JES3 issues to the operator console for disk and tape volumes for a job. Code the FETCH parameter of the JES3 //*MAIN statement to request one of the following:

- All fetch messages for all volumes to be mounted on JES3 setup devices.
- Fetch messages for volumes specified in DD statements that are named in the SETUP parameter on the JES3 //*MAIN statement.
- Fetch messages for volumes on named DD statements.
- No fetch messages.
- No fetch messages for volumes on named DD statements.

Regardless of the FETCH parameter, JES3 sends all the fetch messages to the job log.

Examples

```
//*MAIN  FETCH=ALL
//*MAIN  FETCH=NONE
//*MAIN  FETCH=SETUP
//*MAIN  FETCH=(DDA,INDS,DD7)
//*MAIN  FETCH=/MYDS
```

Messages When Job Exceeds Output Limit

The system sends the operator a warning message when the output from a job exceeds a specified limit. The way you request that the system send a warning message when the limit is exceeded depends on the environment in which your job is executing.

Messages When Output Limit Exceeded in an APPC Scheduling Environment

In an APPC scheduling environment, the BYTES, CARDS, LINES, and PAGES parameters of the JOB statement limit the job's output. When you code the WARNING subparameter with any of these parameters, the system sends the operator a warning message when the output exceeds the limit you have specified.

If you do not code an output limit on the JOB statement BYTES, CARDS, LINES, or PAGES parameter, the system sends a warning message to the operator when a job's output exceeds the installation default limit specified at JES initialization.

Messages When Output Limit Exceeded in a Non-APPC Scheduling Environment

In a non-APPC scheduling environment, you can request that the system send a warning message when the limit is exceeded by using the JOB statement parameters and installation defaults described in Messages When Output Limit Exceeded in an APPC Scheduling Environment. In addition, you can code a BYTES, CARDS, LINES, or PAGES parameter on a JES2 /*JOBPARM statement or on a JES3 //*MAIN statement to limit output for a job.

When you code the WARNING subparameter on the //*MAIN statement, the system sends a warning message to the operator when a job's output exceeds the limit you have specified.

When you code an output limit on the /*JOBPARM statement, the system sends a warning message to the operator when:

- The job's output exceeds the limit you have specified, and
- The warning option has been specified at JES2 initialization as the installation default.

Defaults and Multiple Messages

If you do not code an output limit on the JOB statement, the system uses the limit coded on the //*MAIN statement or the /*JOBPARM statement. If you do not code a //*MAIN or a /*JOBPARM statement, the system uses the installation default limit specified at JES initialization.

If you code multiple //*MAIN statements specifying output limits for a job, or you code a limit and WARNING subparameter on the JOB statement as well as the //*MAIN statement, the operator will receive multiple warning messages.

Use of Warning Messages

One use for the output limit is during program testing. The warning message tells the operator that the program is producing more output than expected. Perhaps the program is in an endless loop that contains instructions sending records to a printer or punch. The operator can halt the program's execution.

Examples

The following examples illustrate the use of the JCL JOB statement, in either an APPC or non-APPC scheduling environment, to warn the operator when the output for a job has exceeded a limit in any JES system:

```
//JOB1  JOB  ACCT01,'D. PIKE',BYTES=(50,WARNING)

//JOB2  JOB  1542,RWALLIN,CARDS=(120,WARNING)

//JOB3  JOB  ,ZOBES,LINES=(200,WARNING)

//JOB4  JOB  ACCT27,'S M SHAY',PAGES=(,WARNING)
```

The following examples illustrate the use of the JES3 //*MAIN statement in a non-APPC scheduling environment to warn the operator when output for a job has exceeded a limit.

```
//*MAIN  BYTES=(50,WARNING)
//*MAIN  CARDS=(120,WARNING)
//*MAIN  LINES=(200,WARNING)
//*MAIN  PAGES=(,WARNING)
```

Communication from System to Userid

The NOTIFY parameter allows the system to notify a user of job or print completion.

Job Completion

When you execute a background or batch job, you can ask the system to notify your time sharing userid or another userid when the job completes. Under TSO/E, a background job is one that is entered from a terminal by a SUBMIT command or by executing a step to run TSO/E in the background. For more information, see *OS/390 TSO/E Command Reference*. A batch job is one that is entered through an input stream.

To request automatic notification, code in your JCL for the job one of the following:

- In a TSO/E background job in a JES2 or JES3 system, specify a userid (and optionally a node) in the JOB statement NOTIFY parameter. If a node is specified, the userid must be attached to that node. If a node is not specified, the userid must be attached to the node from which the job originated.
- In a TSO/E background job or a batch job in a JES2 system, specify a userid in a JES2 /*NOTIFY statement and, if the userid is attached to another node, a node.
- In a batch job in a JES3 system, specify a userid in the JOB statement NOTIFY parameter and the processor for the userid in the ACMAIN parameter of the JES3 //*MAIN statement.

Examples

```
In a JES2 or JES3 system:
     //MYJOB  JOB  ,'I. BUTLER',NOTIFY=DN62PSS
     //MYJOB  JOB  ,'I. BUTLER',NOTIFY=FARNODE.DN62PSS

In a JES2 system:
     /*NOTIFY DN62PSS4
     /*NOTIFY FARNODE.DN62PSS

In a JES3 system:
     //MYJOB  JOB  ,'I. BUTLER',NOTIFY=DN62PSS
     //*MAIN  ACMAIN=2
```

Print Completion

You can receive notification that your output has completed printing by coding the NOTIFY parameter on the OUTPUT JCL statement. NOTIFY allows you to send the print completion message to up to 4 users. The message identifies the output that has completed printing, and indicates whether the printing was successful.

Example

```
//OUT1   OUTPUT  NOTIFY=(PLPSC.ARNOLD,SMYTHE)
```

Communication from Time Sharing Userid to a JES3 System

In a JES3 system, the USER parameter on the JES3 //*MAIN statement identifies the job with a TSO/E user. The job can be submitted through any input source, other than the internal reader, provided the installation does not force job naming conventions. USER allows the TSO/E userid to:

- Issue a TSO/E OUTPUT command to access sysout data sets from the job.
- Inquire about the status of the job or cancel it.

Example

```
//*MAIN USER=J63ET91
```

Communication from Functional Subsystem to Programmer

The programmer can control whether a functional subsystem prints its messages in the output listing following the sysout data set it creates. For this control, code the PIMSG parameter on the OUTPUT JCL statement.

Example

```
//ODS3  OUTPUT PAGEDEF=IMAG4,PIMSG=YES
```

Communication through Job Log

The system produces three system-managed data sets about a job. The system managed-data sets consist of:

- The job log, which is a record of job-related information for the programmer. The job log consists of:

  - The job control statements in the input stream, that is, the JCL statements and JES2 or JES3 statements.
  - Cataloged procedure statements for any procedure a job step calls.
  - Messages about job control statements.
- The job's hard-copy log, which is a record of all message traffic for the job to and from the operator console. These messages describe allocation of devices and volumes, execution and termination of job steps and the job, and disposition of data sets.
- System messages for the job.

The output class for the job log is set by the MSGCLASS parameter on the JOB statement or, if a job-level OUTPUT JCL statement contains a JESDS parameter, by the class that applies to the OUTPUT JCL statement. (**Note:** The MSGCLASS parameter has no effect in an APPC scheduling environment. If you code MSGCLASS, the system will check it for syntax and ignore it.) If no class is specified, the system uses the default class based on the input source of the job; the default is specified at JES initialization.

Printing of the job log is controlled by the following parameters:

- MSGLEVEL parameter of JOB statement.
- All parameters on an OUTPUT JCL statement that contains a JESDS parameter.

To prevent the job log from being printed, code one of the following:

- log subparameter in the JOB statement JES2 accounting information parameter
- NOLOG parameter on the JES2 /*JOBPARM statement

Example 1

```
//JOBC  JOB     ,'V. ST PIERRE',MSGLEVEL=(1,1)
//SMDS  OUTPUT  JESDS=ALL,CLASS=D,COPIES=2,BURST=YES,
```

Example 2

```
//JOBF  JOB     (,,,,,,,N)
/*JOBPARM NOLOG
```

Example 3

```
//J1   JOB    1518,'SECT. E98'
//O1   OUTPUT JESDS=ALL
//O2   OUTPUT JESDS=ALL,WRITER=JCLOGGER
//S1   EXEC   PGM=REPORT
```

This example requests that the three system-managed data sets be printed normally and that a copy of each be routed to an external writer named JCLOGGER.

```
//MYEX     JOB     ,'DEPT. 28H',MSGCLASS=A
//SYSPROG  OUTPUT  JESDS=ALL,GROUPID=SYSPROG
//OPER     OUTPUT  JESDS=ALL,GROUPID=OPER
//USER     OUTPUT  JESDS=ALL,GROUPID=USER,DEFAULT=YES
//REMOTE   OUTPUT  JESDS=ALL,DEST=REMOTE,DEFAULT=YES
//S1       EXEC    PGM=REPORT
//SYSPRINT DD      SYSOUT=A
```

This example creates four different output groups. Group SYSPROG will contain a copy of all three system-managed data sets. Group OPER will also contain a copy of all three system-managed data sets. Group USER will contain a copy of all three system-managed data sets plus a copy of the data set for DD statement SYSPRINT: group USER is processed locally.

The system creates a fourth group with a system-generated group name. This group contains a copy of the three system-managed data sets plus a copy of the data set for DD statement SYSPRINT; this group is processed remotely at destination REMOTE.

Printing Job Log and Sysout Data Sets Together

To print the job log and the sysout data sets from a job on the same output listing, place them in the same output class. Specify one of the following:

- SYSOUT=* on the DD statement.
- CLASS=* on the OUTPUT JCL statement.
- The same output class in the DD SYSOUT parameter or OUTPUT JCL CLASS parameter as specified in the JOB MSGCLASS parameter.

Or, use an OUTPUT JCL statement with a JESDS parameter to control printing of the system-managed data sets. Note that care is needed in specifying the OUTPUT JESDS statement and the sysout DD statement because:

- Any values on the sysout DD statement override those on the OUTPUT JCL statement.
- The values on the OUTPUT JCL statement **always** apply to the system-managed data sets.

Therefore, the output parameters used to process the system-managed output data sets and sysout data sets can be different, even when the data sets all reference the same OUTPUT JCL statement. For example, if the sysout DD statement specifies one output class and the JESDS statement specifies another output class, the sysout data set and system-managed data sets are placed in different subgroups and each is printed in its own output class.

Example 1

```
//J1   JOB  DF16,MSGCLASS=B
//S1   EXEC PGM=ABC
//OUT  DD   SYSOUT=*

//J2   JOB  ,'V. FOTI',MSGCLASS=C
//S1   EXEC PGM=DEF
//OUT  DD   SYSOUT=C

//J3   JOB  ,'G. ROY',MSGCLASS=D
//S1   EXEC PGM=GHI
//OT1  OUTPUT CLASS=*
//DS1  DD   SYSOUT=(,),OUTPUT=*.OT1

//J4   JOB  ,'T. POLAKOWSKI',MSGCLASS=E
//S1   EXEC PGM=JKL
//OT1  OUTPUT DEFAULT=YES,CLASS=E
//DS1  DD   SYSOUT=(,)
```

Example 2

```
//SYSDS   JOB    ,'J. HIGGINS', MSGCLASS=A
//OUT1    OUTPUT JESDS=ALL,GROUPID=JOINT,DEFAULT=YES
//STEP1   EXEC   PGM=REPORT
//REQPRT  DD     SYSOUT=A
```

This example shows how to combine sysout data sets and system-managed output data sets in one output group. The system prints sysout data set REQPRT and all three system-managed data sets in the same group.

Chapter 8. Entering Jobs - Protection

Figure 8-1. Protection Task for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Protection					
through RACF	GROUP PASSWORD SECLABEL USER				

Protection through RACF

The OS/390 Security Server, which includes RACF, is a program product that helps installations achieve data security by controlling the access to data sets and the security level for the execution of jobs. For more information about RACF, see *OS/390 Security Server (RACF) Introduction*.

For RACF protection, the user must supply a userid and a password to RACF. The group name and security label for the job are optional. Depending on the installation's RACF options, the group name and security label can be supplied in the USER, PASSWORD, GROUP, and SECLABEL parameters on the JOB statement. For jobs submitted by a TSO/E user, these items can be obtained from the TSO/E logon.

The security environment of started tasks is defined using a RACF class, not through the USER, PASSWORD, GROUP, and SECLABEL parameters. If these parameters are specified, the started task will fail.

In any RACF installation, the USER and the PASSWORD are required, and the GROUP and the SECLABEL are optional parameters on JOB statements for the following:

- Batch jobs submitted through an input stream, such as a card reader:
  - if the job requires access to RACF-protected resources, or
  - if the installation requires that all jobs have RACF identification.
- Jobs submitted by one RACF-defined user for another user. In this case, the JOB statement must specify the other user's userid and might need a password. The group id and security label are optional.
- Jobs that execute at another network node that uses RACF protection.

Examples

```
//MYJOB  JOB  D58,SUE,USER=D58STW,PASSWORD=41168X
//YOURS  JOB  D58,DON,USER=DSCHOF,PASSWORD=404632,GROUP=D58DISK
//RAJOB  JOB  D58,ALE,USER=D59AFG,PASSWORD=3316YX,SECLABEL=CONF
```

© Copyright IBM Corp. 1988, 1999

Chapter 9. Entering Jobs - Resource Control

Figure 9-1. Resource Control Task for Entering Jobs

TASKS FOR ENTERING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Resource Control					
of program library			JOBLIB DD STEPLIB DD DD defining member of PDS or PDSE		
of procedure library			JCLLIB	PROCLIB on /*JOBPARM	PROC and UPDATE on //*MAIN
of INCLUDE group			JCLLIB	PROCLIB on /*JOBPARM	PROC and UPDATE on //*MAIN
of address space	REGION ADDRSPC	REGION ADDRSPC			LREGION on //*MAIN
of processor	SCHENV			SYSAFF on /*JOBPARM	SYSTEM on //*MAIN
of spool partition					SPART and TRKGRPS on //*MAIN

Resource Control of Program Library

To be executed, a program must be in one of the following libraries:

System library
Private library
Temporary library

A library is a partitioned data set (PDS) or a partitioned data set extended (PDSE) on direct access storage. PDSs and PDSEs are divided into partitions, called members. In a library, each member contains a program or part of a program.

For details on creating and deleting members in a PDS or PDSE, see *DFSMS/MVS Using Data Sets*.

System Library

Unless a job or step specifies a private library, the system searches for a program in the system libraries when you code:

```
//stepname  EXEC  PGM=program-name
```

The system looks in the libraries for a member with a name or alias that is the same as the specified *program-name*. The most used system library is SYS1.LINKLIB, which contains executable programs that have been processed by the linkage editor.

© Copyright IBM Corp. 1988, 1999

If an earlier DD statement in the job defines the program as a member of a system library, refer to that DD statement to execute the program:

```
//stepname  EXEC  PGM=*.stepname.ddname
```

Private Library

Each executable, user-written program is a member of a private library. To tell the system that a program is in a private library, code a DD statement defining that library in one of the following ways:

- To define a private library to be used throughout a job, place a DD statement with the ddname JOBLIB after the JOB statement and before the first EXEC statement in the job.
- To define a library to be used in only one step, place a DD statement with the ddname STEPLIB in the step.

To execute a program from a private library, code:

```
//stepname  EXEC  PGM=program-name
```

When you code JOBLIB or STEPLIB, the system searches for the program to be executed in the library defined by the JOBLIB or STEPLIB DD statement before searching in the system libraries.

If an earlier DD statement in the job defines the program as a member of a private library, refer to that DD statement to execute the program:

```
//stepname  EXEC  PGM=*.stepname.ddname
```

Use of Private Libraries

Private libraries are particularly useful for programs used too seldom to be needed in a system library. For example, programs that prepare quarterly sales tax reports are good candidates for a private library.

Creating a Private Library

To create a private library, code a JOBLIB or STEPLIB DD statement and add one or more members to it in the job. The JOBLIB library is more convenient than the STEPLIB, because the JOBLIB is available to every step in the job in order to add members or to execute already added members. The STEPLIB DD must be passed or redefined in each step that uses it.

Adding Members to a Private Library

To add members to a library, code a DD statement that defines the library and names the member to be added to the library.

Example of Creating and Adding to a Private Library

```
//EG      JOB  5328,'MARGARET NONNSEN'
//JOBLIB  DD   DSNAME=GROUPLIB,DISP=(NEW,CATLG),
//           UNIT=3350,VOL=SER=727104,
//           SPACE=(CYL,(50,3,4))
//STEP1   EXEC PGM=FIND
//ADDPGMD DD   DSNAME=GROUPLIB(RATE),DISP=MOD,
//           VOL=REF=*.JOBLIB
//STEP2   EXEC PGM=RATE
```

In this example, the JOBLIB DD statement creates a library named GROUPLIB. Program FIND in STEP1 adds the program RATE to the library. STEP2 calls the program RATE.

In STEP1, the system looks for the program named FIND in SYS1.LINKLIB, because the private library created on the JOBLIB DD statement does not actually exist until a member is added to it. In STEP2, the system looks for the program named RATE first in the JOBLIB library.

Retrieving an Existing Private Library

If several programs for a job are in the same private library, identify the library on a JOBLIB DD statement. The library is available in every step of the job for which you do not code a STEPLIB DD statement.

To make a library available to a single step, identify the library on a STEPLIB DD statement. The STEPLIB library is available only to the step that contains the STEPLIB DD statement, unless you pass the library and retrieve it in a subsequent step.

The system searches for a program in the private library you identify. If a job contains a JOBLIB DD statement and a step contains a STEPLIB DD statement, the system searches for the step's program first in the STEPLIB library and then in the system libraries. The system ignores the JOBLIB library for that step.

For a step in a job using a JOBLIB library, if you want the system libraries searched rather than the JOBLIB, code a STEPLIB DD statement that identifies a system library:

```
//STEPLIB DD  DSNAME=SYS1.LINKLIB,DISP=SHR
```

Example of Retrieving Job and Step Libraries

```
//MYJOB   JOB  MSGLEVEL=1
//JOBLIB  DD   DSNAME=LIB5.GRP4,DISP=SHR
//STEP1   EXEC PGM=FIND
//STEP2   EXEC PGM=GATHER
//STEPLIB DD   DSNAME=ACCOUNTS,DISP=(SHR,KEEP),
//           UNIT=3350,VOL=SER=727104
```

- In STEP1, the system searches the library named LIB5.GRP4, defined on the JOBLIB DD statement, for the program named FIND.
- In STEP2, the system searches the library named ACCOUNTS, defined on the STEPLIB DD statement, for the program named GATHER.

Concatenating Private Libraries

If a job uses programs from several libraries, you can concatenate these libraries to a JOBLIB DD statement or a STEPLIB DD statement; all the libraries being concatenated must be existing libraries. Omit the ddname from all the DD statements for the libraries, except the first.

The system searches the libraries for the program in the same order as the DD statements.

Example of Concatenated Libraries

```
//JOBLIB DD  DSNAME=D58.LIB12,DISP=(SHR,PASS)
//       DD  DSNAME=D90.BROWN,DISP=(SHR,PASS),
//         UNIT=3330,VOL=SER=411731
//       DD  DSNAME=A03.EDUC,DISP=(SHR,PASS)
```

Temporary Library

Temporary libraries are partitioned data sets created to store a program until it is used in a later step *of the same job*. A temporary library is created and deleted within a job.

When testing a newly written program, a temporary library is particularly useful for storing the load module from the linkage editor until it is executed by a later job step. Because the module will not be needed by other jobs until it is fully tested, it should not be stored in a system library.

While the system assigns the module a name in the temporary library, the name cannot be predicted. Therefore, use the PGM parameter to identify the program by location rather than by name. Code a backward reference to the DD statement that defines the temporary library:

```
//stepname  EXEC  PGM=*.stepname.ddname
```

Creating a Temporary Library

In the step that produces the program, code a DD statement that creates a partitioned data set and place the program in it. A later step can then retrieve this program. Alternatively, you can use the virtual I/O (VIO) facilities to define a temporary library. See “Allocation of Virtual I/O” on page 15-49 for details.

Example

```
//STEP2    EXEC PGM=IEWL
             .
             .
             .
//SYSLMOD  DD   DSNAME=&&PARTDS(PROG),UNIT=3350,
//           DISP=(NEW,PASS),SPACE=(1024,(50,20,1))
//STEP3    EXEC PGM=*.STEP2.SYSLMOD
```

STEP2 calls the program IEWL, which link edits object modules to form a load module that can be executed. STEP2 places the module in the library defined in the SYSLMOD DD statement.

STEP3 calls the program by naming the step that created the library and the DD statement that defines the program as a member of a library. If STEP2 had called a

procedure and the DD statement named SYSLMOD was included in PROCSTEP3 of the procedure, you would code PGM=*.STEP2.PROCSTEP3.SYSLMOD.

Resource Control of Procedure Library

Procedure libraries are partitioned data sets consisting of members that contain procedures or INCLUDE groups. For information about INCLUDE groups, see “Resource Control of INCLUDE Group” on page 9-6.

To call and execute a procedure cataloged in a library, code:

```
//stepname EXEC  PROC=procedure-name
```

The name of the cataloged procedure is its member name or alias in the library.

Retrieving a Procedure Library

If a job does not specify a procedure library, the system retrieves all cataloged procedures called by EXEC statements from the procedure libraries defined by the installation for the job's job class.

If a job's cataloged procedures are contained in another procedure library, use the following parameters to direct the system to that library. The parameters must specify procedure libraries defined during JES initialization.

- Code a JCLLIB statement to tell the system to search system procedure libraries, installation-defined procedure libraries, or private libraries. The system searches the libraries in the order in which they are specified on JCLLIB.
- In a JES2 system, code a PROCLIB parameter on the JES2 /*JOBPARM statement.
- In a JES3 system, code a PROC parameter on the JES3 //*MAIN statement.

Updating a Procedure Library

To add a procedure to an installation-defined procedure library or to modify permanently a procedure in a library, use the IEBUPDTE utility program. If modifying, tell the system operator to delay any jobs that would use the procedure during modification.

In a JES3 system, you can specify UPDATE on the JES3 //*MAIN statement to update a procedure library. This parameter causes all jobs using the identified data set and any concatenated data sets to be held until the update is complete.

Examples

```
//JOB1   JOB
//LIBS   JCLLIB  ORDER=(MYPRI.PROCS.JCL,SYS1.PROCLIB,INSTALL.JCL.PROCS)
//STEP1  EXEC    PROC=STAT
          .
          .
          .
```

In a JES2 system:

```
//JOB87  JOB  ,'S. WENDALL'
/*JOBPARM PROCLIB=PROC15
//S1     EXEC  PROC=ALEG
//INDS   DD    *
         .
        (data)
         .
/*
```

In a JES3 system:

```
//JOB87  JOB  ,'S. WENDALL'
//*MAIN PROC=15
//S1     EXEC  PROC=ALEG
//INDS   DD    *
         .
        (data)
         .
/*
```

In these examples, the system obtains the procedure ALEG from the procedure library PROC15.

Resource Control of INCLUDE Group

An INCLUDE group is a member of a system library, installation-defined library, or private library.

To imbed an INCLUDE group in the JCL stream at the point of the INCLUDE statement, code:

```
//name INCLUDE MEMBER=member-name
```

The system replaces the INCLUDE statement with the JCL statements contained in the INCLUDE group.

Retrieving an INCLUDE Group

To tell the system to search system libraries, installation-defined libraries, or private libraries for the member named on an INCLUDE statement, code:

```
//name JCLLIB ORDER=library-name1,library-name2
```

Example

```
//IDLIB     JCLLIB  ORDER=(PRILIB.INCL.ONE,PRILIB.INC.TWO)
//INCGRP    INCLUDE MEMBER=OUTSTMTS
```

Resource Control of Address Space

Types of Storage

In MVS, the storage available for a program is virtual storage or central storage (also called real storage):

- **Virtual storage** is addressable space that appears to the user as central (real) storage. Instructions and data are mapped from virtual storage into central storage locations, where they are executed.
- **Central (real) storage** is the storage from which the processor can directly obtain instructions and data and to which it can directly return results.

Virtual Storage

The virtual storage address space is 2 gigabytes. The address space contains the commonly addressable system storage, the nucleus, and the private address space, which includes the user's region.

When a program is selected, the system brings it into virtual storage and divides it into pages of 4K bytes. The system transfers the pages of a program into central (real) storage for execution and out to auxiliary storage when not needed. Paging is done automatically; to the programmer, the entire program appears to occupy contiguous space in central storage at all times. Actually, not all pages of a program are necessarily in central storage at one time. Also, the pages that are in central storage do not necessarily occupy contiguous space.

Central (Real) Storage

Certain programs must have all their pages in contiguous central (real) storage while they are executing. They cannot be paged. These programs must be put into an area of virtual storage called the nonpageable dynamic area, whose virtual addresses are identical to real addresses.

Such programs include:

- Programs that modify a channel program while it is active.
- Programs that are highly dependent on time.

Such programs are the only ones for which you should request central storage. To request central storage, code ADDRSPC=REAL on the JOB or EXEC statement and request the amount of central storage needed in a REGION parameter.

Requesting Amount and Type of Storage

The amount of space needed by a job or step can be specified in the REGION parameter of the JOB or EXEC statement. If REGION is on the JOB statement, each step of the job executes in the requested amount of space. If on the EXEC statements in a job, each step executes in its own amount of space. Use the EXEC statement REGION parameters when different steps need greatly different amounts of space.

The REGION parameter differs depending on whether the program uses virtual or central storage.

Region Size for Virtual Storage

When ADDRSPC=VIRT is coded or implied, the system establishes two values from the REGION parameter or the installation-defined default. These values are:

- An upper boundary to limit region size for variable-length GETMAINs.
- A second limiting value set by the IBM- or installation-supplied routine IEALIMIT or IEFUSI. The system uses this second value to limit:
  - Fixed-length GETMAINs.
  - Variable-length GETMAINs when the space remaining in the region is less than the requested minimum.

  When the minimum requested length for a variable-length GETMAIN or the amount requested for a fixed-length GETMAIN exceeds this second value, the job or step abnormally terminates. See *OS/390 MVS Initialization and Tuning Guide* and *OS/390 MVS Programming: Assembler Services Guide* .

The amount of space requested must include the following:

- Space for all programs to be executed.
- All additional space the programs request with GETMAIN macro instructions during execution.
- Enough unallocated space for task termination.

Region Size for Central (Real) Storage

When ADDRSPC=REAL is coded, the system establishes one value from the REGION parameter or the installation-defined default. The value is used as an upper boundary to limit region size for all GETMAINs.

The minimum region size must be:

- 8K if the program to be executed is reenterable and resides in an authorized library.
- 12K for all other programs.

Note that this is the minimum region for successful execution, but not necessarily the minimum region size for successful job completion. Programs executed in central storage should perform as much clean-up as possible before terminating.

Example 1

```
//J28  JOB  ,'F. GOLAZESKI',CLASS=D
//S1   EXEC PGM=PROGREAL,REGION=20K,ADDRSPC=REAL
//DD1  DD   DSNAME=A.B.C,DISP=OLD
//S2   EXEC PGM=PROGVIRT,REGION=75K,ADDRSPC=VIRT
//DD2  DD   DSNAME=MYDS2,DISP=OLD
```

This example shows how to request storage for a program that must not be paged and for a program that can be paged. Step S1 executes in central (real) storage, without paging, while step S2 executes in virtual storage, with paging.

Example 2

```
//STEPA  EXEC  PROC=MYPROC8,REGION.FIRST=750K,
//          REGION.SECOND=700K
```

This EXEC statement assigns space requests to two procedure steps, FIRST and SECOND, of a procedure named MYPROC8.

OS/390 UNIX System Services Considerations

In OS/390 UNIX System Services, callable service BPX1SRL lets a program modify its REGION size. Note that only superusers can increase their REGION size. See *OS/390 UNIX System Services Programming: Assembler Callable Services Reference* for more information on the BPX1SRL callable service.

Requesting Amount of Logical Storage in a JES3 System

The LREGION parameter of the JES3 //*MAIN statement allows you to specify the approximate size of the largest step's working set in central (real) storage. JES3 uses the LREGION value to improve job scheduling. For more information, see *OS/390 JES3 Initialization and Tuning Reference*.

Use LREGION carefully. If the values selected for LREGION are too small, the job may take longer to run.

Example

```
//*MAIN LREGION=100K
```

Resource Control of the Processor

Selecting a Processor Using A Scheduling Environment

You can specify the name of a WLM scheduling environment, using the SCHENV parameter on the JOB statement. A scheduling environment is a list of resources and their required settings. By associating a scheduling environment name with a job, you ensure that the job will be scheduled only on a system that satisfies those resource state requirements.

Scheduling environments differ from the JES2 SYSAFF and JES3 SYSTEM statements (presented in the next sections). A scheduling environment is abstract and dynamic. It identifies the dependency that a job has to run on particular systems without specifically naming the systems. Since a scheduling environment can change state, the systems where a job is eligible to run can change without modification to its JCL. The SYSAFF and SYSTEM statements are specific and static, since they list system names.

Also, the SYSAFF statement controls where a job converts and executes, whereas a scheduling environment controls only where a job executes. (The SYSTEM statement does not differ from a scheduling environment in this way — both control only where a job executes.)

You can use scheduling environments and the SYSAFF or SYSTEM statement together. A job may be restricted to either SYS1 or SYS2, for instance, based on the scheduling environment associated with that work. The SYSAFF or SYSTEM statement may then further restrict that work only to SYS1.

For more information about WLM scheduling environments, see *OS/390 MVS Planning: Workload Management*.

Example

```
//JOBA  JOB  1,'STEVE HAMILTON',SCHENV=DB2LATE
```

Selecting a Processor in JES2

In a JES2 multi-access spool configuration, jobs enter from local input streams, from remote work stations, and from processors at other network nodes. If an entering job does not specify a system, JES2 can assign the job to execute on any system in the configuration.

In a multi-access spool configuration, a job can request execution on specific systems. This request is made by coding:

```
/*JOBPARM SYSAFF=cccc
/*JOBPARM SYSAFF=(cccc,cccc,cccc)
/*JOBPARM SYSAFF=*
/*JOBPARM SYSAFF=ANY
```

A specified system processes the job's JCL and executes the job. The output from the job can be processed by any system in the multi-access spool configuration.

You should request a specific system when a job has special processing requirements not available on all systems in the configuration. For example, an emulation job might need to run on a particular system.

For more information on the JES2 multi-access spool configuration, see *OS/390 JES2 Initialization and Tuning Guide*.

Independent Mode

If the job needs to be processed by a system in independent mode, code:

```
/*JOBPARM SYSAFF=(cccc,IND)
/*JOBPARM SYSAFF=(,IND)
/*JOBPARM SYSAFF=(ANY,IND)
```

A specified system, provided it is operating in independent mode, processes the job's JCL and executes the job. The same system processes the job's output.

Independent mode is useful for testing new components with selected jobs while in a shared configuration.

Examples

```
/*JOBPARM SYSAFF=SYS2
/*JOBPARM SYSAFF=(S333,IND)
/*JOBPARM SYSAFF=(*,IND)
```

Selecting a Processor in JES3

JES3 automatically selects a processor for a job based on the resources that JES3 knows the job needs in order to execute. These resources are:

- Devices
- Volumes
- Data sets

- Processor features, such as an emulator, a nonstandard catalog, or a connection to a particular system-managed device.

If a job must have resources that JES3 does not control or that JES3 cannot infer from the job control statements, name the processor(s) that should or should not execute the job by coding:

```
//*MAIN SYSTEM=ANY
//*MAIN SYSTEM=JGLOBAL
//*MAIN SYSTEM=JLOCAL
//*MAIN SYSTEM=(main-name,main-name,...)
//*MAIN SYSTEM=/(main-name,main-name,...)
```

Relationship to Other Parameters

The requested processor must be consistent with other parameters specified in the job control statements:

- CLASS parameter on the JOB statement or //*MAIN statement. A processor or processors are defined for each valid job class during JES3 initialization. If the SYSTEM parameter specifies a processor that does not execute jobs of the specified class, JES3 abnormally terminates the job.
- DD statement UNIT parameter that specifies a device-number for a device that is JES3-managed or jointly JES3/MVS managed. The specified device must be attached to the requested processor. Also, because a specific device is requested, the SYSTEM parameter is required.
- The TYPE parameter on the //*MAIN statement must specify the system running on the requested processor.
- The processing requests made in JES3 //*PROCESS statements. Any dynamic support programs called in //*PROCESS statements must be able to be executed on the requested processor.

Examples

```
//*MAIN SYSTEM=(PRS1,PRS3)
```

Resource Control of Spool Partitions in a JES3 System

When JES3 reads a job, it initially places the job on a spool volume or volumes. The spool volumes can be divided by the installation into groups, known as partitions. During JES3 initialization, partitions are defined and associated with output classes, job classes, and processors. See *OS/390 JES3 Initialization and Tuning Guide* for details.

During job processing, JES3 allocates spool data sets to a partition, as follows, in override order:

1. The spool partition for the output class of the sysout data set.
2. The spool partition for the job's class.
3. The spool partition for the processor executing the job.
4. The default spool partition.

You can use the //*MAIN statement to override the JES3 partition allocations, except for allocation of partitions for sysout data sets and SYSIN data sets. A sysout data set is always placed in the partition used for its output class; a SYSIN

data set is always placed in the default spool partition. Depending on how the installation defines the partitions, you can make JES3 allocate all the spool data for a job or all the spool data of a particular type, such as output, to a specified spool partition. Thus, you can limit the number of spool volumes that JES3 uses for a job's spool data sets. To control the spool partition, code:

```
//*MAIN SPART=partition-name
```

Example 1

```
//ONE   JOB   ,'PAT EGAN'
//*MAIN SYSTEM=SY2
//S1    EXEC  PGM=ABC
//OUT1  DD    SYSOUT=N
//OUT2  DD    SYSOUT=S
```

During initialization, the installation assigned spool partitions as follows:

- PARTD is assigned to output class S.
- PARTC is assigned to processor SY2.
- PARTA is the default partition.
- No partition is assigned to output class N.

The job's input spool data sets are allocated to the default spool partition, PARTA.

Because the job executes on processor SY2 and no partition is assigned for output class N, the sysout data set OUT1 is allocated to partition PARTC.

Sysout data set OUT2 is allocated to PARTD.

Example 2

```
//TWO   JOB    ,'LEE BURKET'
//*MAIN CLASS=IMSBATCH,SYSTEM=SY2
//S1    EXEC   PGM=DEF
//OUT1  DD     SYSOUT=N
//OUT2  DD     SYSOUT=S
```

During initialization, the installation assigned spool partitions as for job ONE, with the following addition:

- PARTB is assigned to job class IMSBATCH.

The sysout data set OUT1 is allocated to partition PARTB, the job class's partition. Note that the job class's partition overrides the processor's partition.

Example 3

```
//THREE JOB    ,'T. POLAKOWSKI'
//*MAIN CLASS=IMSBATCH,SPART=PARTE,SYSTEM=SY2
//STEP1 EXEC   PGM=GHI
//OUT   DD     SYSOUT=N
//OUT2  DD     SYSOUT=S
```

During initialization, the installation assigned spool partitions as for job TWO.

The sysout data set OUT1 is allocated to partition PARTE, as specified in the SPART parameter. Note that the SPART parameter overrides the processor's partition and the job class's partition.

Part 3. Tasks for Processing Jobs

Chapter 10. Processing Jobs - Processing Control 10-1
Processing Control by Conditional Execution 10-1
Bypassing or Executing Steps Based on the Evaluation of Previous Steps 10-1
Using the IF/THEN/ELSE/ENDIF Statement Construct 10-2
Uses of Return Code Tests 10-2
Job and Step Level Evaluation Using the IF/THEN/ELSE/ENDIF Statement Construct 10-3
Relationship of the IF/THEN/ELSE/ENDIF Statement Construct to the COND Parameter 10-4
Step Execution After a Preceding Step Abnormally Terminates 10-4
Steps that Do Not Execute after A Preceding Step Abnormally Terminates 10-5
Examples of IF/THEN/ELSE/ENDIF Statement Construct 10-5
Bypassing or Executing Steps Based on Return Codes 10-6
Uses of Return Code Tests 10-6
Relationship of the COND Parameters on JOB and EXEC Statements 10-7
Step Execution after a Preceding Step Abnormally Terminates 10-7
Examples of JOB Statement Return Code Tests 10-8
Examples of EXEC Statement Return Code Tests 10-9
Examples of EXEC COND Parameters with EVEN and ONLY 10-9
Examples of COND Return Code Testing in a Job 10-10
Examples of COND Parameters in Procedures 10-12
Examples of COND Parameters that Force Step Execution 10-13
Processing Control by Cancelling a Job that Exceeds Output Limit 10-13
Limiting Output in an APPC Scheduling Environment 10-13
Limiting Output in a Non-APPC Scheduling Environment 10-14
Use in Testing 10-14
Processing Control by Timing Execution 10-15
JOB and EXEC TIME Parameter 10-15
JES2 Time Parameters 10-16
OS/390 UNIX System Services Considerations 10-16
Processing Control for Testing 10-16
Altering Usual Processing for Testing 10-16
Scanning JCL for Errors (Non-APPC) 10-17
Using IEFBR14 Program for Testing 10-17
Using Nonstandard Processing 10-18
Dumping after Error 10-19

Chapter 11. Processing Jobs - Performance Control 11-1
Performance Control by Job Class Assignment (Non-APPC) 11-1
Performance Control by Selection Priority (Non-APPC) 11-2
Priority for JES2 Jobs 11-2
Priority for JES3 Jobs 11-3
Priority Aging 11-3
Performance Control by Performance Group (Non-APPC) 11-3
Performance Control by I/O-to-Processing Ratio (Non-APPC) 11-4

This part describes how to process jobs that have been entered into the system. These tasks are all optional. They are:

- Processing control

© Copyright IBM Corp. 1988, 1999

- Performance control

Chapter 10. Processing Jobs - Processing Control

Figure 10-1. Processing Control Task for Processing Jobs

TASKS FOR PROCESSING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Processing Control					
by conditional execution	COND CANCEL on BYTES, CARDS, LINES, and PAGES	COND	IF/THEN/ELSE/ENDIF statement construct	CANCEL on BYTES, CARDS, LINES, and PAGES on /*JOBPARM	CANCEL on BYTES, CARDS, LINES, and PAGES on //*MAIN
by timing execution	TIME or time in JOB JES2 accounting information	TIME		TIME on /*JOBPARM	
for testing: 1. by altering usual processing 2. by dumping after error	TYPRUN CLASS DUMP on BYTES, CARDS, LINES, and PAGES	PGM=IEFBR14 PGM=JCLTEST PGM=JSTTEST (JES3 only)	SYSMDUMP DD SYSUDUMP DD SYSABEND DD To format dump on 3800 Printing Subsystem, FCB=STD3 and CHARS=DUMP on dump DD.		//*PROCESS //*ENDPROCESS DUMP on BYTES, CARDS, LINES, and PAGES on //*MAIN

Processing Control by Conditional Execution

You can conditionally execute steps in a job by using the IF/THEN/ELSE/ENDIF statement construct or the COND parameter.

Bypassing or Executing Steps Based on the Evaluation of Previous Steps

Depending on the results of a job step, you might need to bypass or execute later steps. For instance, if a step terminates abnormally, you might want to execute an error routine procedure; while if the step terminates normally, you want to continue processing with the next step.

© Copyright IBM Corp. 1988, 1999

Using the IF/THEN/ELSE/ENDIF Statement Construct

You can conditionally execute job steps with the IF/THEN/ELSE/ENDIF statement construct. Use this statement construct instead of the COND parameter to conditionally execute job steps based on:

- Return codes
- Abend conditions
- System or user abend completion codes.

The IF/THEN/ELSE/ENDIF statement construct tests whether these conditions occurred in the job, a step, or a procedure step prior to the IF/THEN/ELSE/ENDIF statement construct.

You can code the IF/THEN/ELSE/ENDIF statement construct anywhere in the job after the JOB statement. Code it as follows:

```
//[name]     IF   (relational expression)   THEN
//STEPTRUE   EXEC
//[name]     ELSE
//STEPFALS   EXEC
//                      ENDIF
```

The relational expression consists of:

- Comparison operators
- Logical operators
- Not (¬) operators
- Relational expression keywords.

Comparison operators compare a relational expression keyword to a numeric value. The comparison results in a true or false condition. Use the logical operators & (AND) and | (OR) in complex relational expressions, to indicate that the system evaluates the Boolean result of two or more relational expressions. The ¬ (NOT) operator reverses the testing of the relational expression. Relational expression keywords indicate that you are evaluating a return code, abend condition, or abend completion code.

Either the THEN clause or ELSE clause must contain at least one EXEC statement. The EXEC statement indicates a job step that the system executes based on its evaluation of the relational expression. A THEN or ELSE clause that does not contain an EXEC statement is a null clause.

You can nest IF/THEN/ELSE/ENDIF statement constructs up to 15 levels of nesting.

Uses of Return Code Tests

Certain IBM programs produce standard return codes. For example, a compiler or linkage editor returns a code of 8 to indicate serious errors in the compiled or link-edited program; the program may not execute correctly. Before executing a newly compiled or link-edited program, test the return code from the compiler or linkage editor; if it is 8, bypass execution of the program.

In user-written programs, assign a return code to signify a certain condition. For example, STEP1 of a job reads accounts that subsequent steps process. STEP1

sets a return code of 10 if delinquent accounts are found. STEP3 processes only delinquent accounts. Before STEP3 executes, test the return code from STEP1:

- If the return code from STEP1 is 10, indicating delinquent accounts, execute STEP3.
- If the return code from STEP1 is not 10, bypass STEP3.

Code the IF/THEN/ELSE/ENDIF statement construct as follows:

```
//RCTEST   IF   (STEP1.RC = 10)   THEN
//STEP3    EXEC
//IFNOT    ELSE
//         ENDIF
//NEXTSTEP EXEC
```

Compatible Return Code Tests: The system applies the return code tests on the IF/THEN/ELSE/ENDIF statement construct to the return code, if any, produced by a job, step, or procedure step in the job. To take advantage of this statement construct, the return codes for each step should have compatible meanings. For example, the COBOL compiler and the linkage editor have compatible return codes:

4 Minor errors were found, but a compiled program or load module was produced. Execution may be successful.

8 Major errors were found, but a compiled program or load module was produced. Execution will probably not be successful.

12 Serious errors were found. A compiled program or load module was not produced.

To continue processing in spite of small errors, code the return code test as follows:

```
//NOTBAD    IF   (RC > 4)   THEN
//BADERR    EXEC   PGM=ERRRTN
//NOGOOD    ELSE
//NEXTSTEP  EXEC
//          ENDIF
```

When a previous job step has a return code greater than 4, step BADERR executes an error routine procedure called ERRRTN. When the return code on all previous job steps is less than or equal to 4, the ELSE statement allows processing to continue with step NEXTSTEP.

Job and Step Level Evaluation Using the IF/THEN/ELSE/ENDIF Statement Construct

The way you code the IF/THEN/ELSE/ENDIF statement construct determines whether the statement construct tests all job steps, a single job step, or a procedure step.

Job Level Evaluation: If you do not code a stepname, the IF/THEN/ELSE/ENDIF statement construct evaluates the return code, abend condition, or run condition of every previous step in the job. If the condition (return code, abend condition, or run condition) is satisfied based on the steps in the job that have executed thus far, the system executes the THEN clause.

Step Level Evaluation: To test a single step, code the **stepname** of the step you want to test. To test a procedure step, code the **stepname.procstepname** of the procedure step you want to test. If the step or procedure step that you are evaluating did not execute, was cancelled or ended abnormally, the result of the evaluation is false.

Relationship of the IF/THEN/ELSE/ENDIF Statement Construct to the COND Parameter

When you specify both the IF/THEN/ELSE/ENDIF statement construct and the COND parameter for a job step, the job step represented by the EXEC statement will execute only when both the IF/THEN/ELSE/ENDIF statement construct and the COND parameter evaluate to execute.

If a job abends and no abend condition was specified on the IF/THEN/ELSE/ENDIF statement construct or the COND parameter, the default is that a job step will not execute. When both the IF/THEN/ELSE/ENDIF statement construct and the COND parameter are specified for a job step, the default is applied only when neither specifies an abend condition.

The system evaluates a COND parameter on the first EXEC statement in a job as false. However, you can use an IF statement before the first EXEC statement in a job to bypass the step.

Step Execution After a Preceding Step Abnormally Terminates

Abnormal termination of a step usually causes the system to bypass subsequent steps and to terminate the job. However, the IF/THEN/ELSE/ENDIF statement construct lets you request execution of a step when a previous step terminates abnormally.

Testing for an Abend Condition: When a job step abends, the system scans the remaining steps for an IF/THEN/ELSE/ENDIF statement construct that tests for an abend or abend completion code. If none is present, the system terminates the job.

Code one of the following to execute an error routine program after an abend:

```
//IFBAD      IF    (ABEND)   THEN
//ERROR      EXEC  PGM=ERRRTN
//           ENDIF
//NEXTSTEP   EXEC

              or:

//IFBAD      IF    (ABEND=TRUE)   THEN
//ERROR      EXEC   PGM=ERRRTN
//           ENDIF
//NEXTSTEP   EXEC
```

The system executes step ERROR only when one or more of the preceding steps abnormally terminates.

Testing for an Abend Completion Code: To execute a step based on the evaluation of an abend completion code, code:

```
//IFABEND    IF    (ABENDCC=S0C4)   THEN
//ABNDSTEP   EXEC  PGM=CLEANUP
//           ENDIF
//NEXTSTEP   EXEC
```

The system executes the program CLEANUP when a previous step has the system abend completion code 0C4.

Steps that Do Not Execute after A Preceding Step Abnormally Terminates

Certain error conditions prevent the system from executing the THEN or ELSE clauses of an IF/THEN/ELSE/ENDIF statement construct. When one of these error condition occurs, the system does not execute the THEN or ELSE clause, regardless of any tests on the IF statement. Such errors conditions occur when:

- Certain system completion codes are issued
- Job time expires
- A referenced data set is not complete
- The program does not have control.

For more information about errors that prevent execution regardless of IF statement tests, see *OS/390 MVS JCL Reference*.

Examples of IF/THEN/ELSE/ENDIF Statement Construct

Example 1: This example tests the return code for a step.

```
//RCTEST     IF    (STEP1.RC GT 20|STEP2.RC = 60)   THEN
//STEP3      EXEC PGM=U
//ENDTEST    ENDIF
//NEXTSTEP   EXEC
```

The system executes STEP3 if

- The return code from STEP1 is greater than 20, or
- The return code from STEP2 equals 60.

If the evaluation of the relational expression is false, the system bypasses STEP3 and continues processing with step NEXTSTEP.

Example 2: This example tests for an abend condition in a procedure step.

```
//ABTEST     IF     (STEP4.LINK.ABEND=FALSE)   THEN
//BADPROC    ELSE
//CLEANUP    EXEC   PGM=ERRTN
//ENDTEST    ENDIF
//NEXTSTEP   EXEC
```

The relational expression tests that an abend did not occur in procedure LINK, called by the EXEC statement in STEP4. If the relational expression is true, no abend occurred. The null THEN statement passes control to step NEXTSTEP. If the relational expression is false, an abend occurred. The ELSE clause passes control to the program called ERRTN.

Example 3: This example tests for a user abend completion code in the job.

```
//CCTEST   IF   (ABENDCC = U0100)   THEN
//GOAHEAD  EXEC  PGM=CONTINUE
//NOCC     ELSE
//EXIT     EXEC  PGM=CLEANUP
//         ENDIF
```

If any job step produced the user abend completion code 0100, the EXEC statement GOAHEAD calls the procedure CONTINUE. If no steps produced the completion code, the EXEC statement EXIT calls program CLEANUP.

Bypassing or Executing Steps Based on Return Codes

To indicate the results of its execution, a program can issue a *return code*. Using a COND parameter, you can test the return code and, based on the test, either bypass or execute a step.

The COND parameter can be specified on either a JOB or EXEC statement by coding:

```
//jobname JOB acct,progname,COND=(code,operator)
//jobname JOB acct,progname,COND=((code,operator),(code,operator))

//stepname EXEC PGM=x,COND=(code,operator)
//stepname EXEC PGM=x,COND=(code,operator,stepname)
//stepname EXEC PROC=x,COND=((code,operator,stepname.procstepname))

//stepname EXEC PGM=x,COND=EVEN
//stepname EXEC PGM=x,COND=ONLY
//stepname EXEC PGM=x,COND=((code,operator),EVEN)
//stepname EXEC PGM=x,COND=((code,operator,stepname),ONLY)
```

If an EXEC statement COND parameter causes a step to be bypassed, only that step is not executed; the following steps are executed or not, depending on their COND parameters. If a JOB statement COND parameter causes a step to be bypassed, the system bypasses all remaining job steps.

Bypassing a step because of an EXEC COND parameter is not the same as abnormally terminating the step. Bypassing permits the following steps to be executed; abnormally terminating causes all following steps to be bypassed, unless they contain EVEN or ONLY in their EXEC COND parameters.

Uses of Return Code Tests

Certain IBM programs produce standard return codes. For example, a compiler or linkage editor returns a code of 8 to indicate serious errors in the compiled or link-edited program; the program may not execute correctly. Before executing a newly compiled or link-edited program, test the return code from the compiler or linkage editor; if it is 8, bypass execution of the program.

In user-written programs, assign a return code to signify a certain condition. For example, STEP1 of a job reads accounts that subsequent steps process. STEP1 sets a return code of 10 if delinquent accounts are found. STEP3 processes only delinquent accounts. Before STEP3 executes, test the return code from STEP1:

- If the return code from STEP1 is 10, indicating delinquent accounts, execute STEP3.
- If the return code from STEP1 is not 10, bypass STEP3.

Relationship of the COND Parameters on JOB and EXEC Statements

The effect of return code tests on the different statements is:

- **The JOB statement COND parameter** performs the same return code tests for every step in a job. If a JOB statement return code test is satisfied, the job terminates.
- **An EXEC statement COND parameter** performs return code tests for only its step in a job. Using EXEC COND parameters, different tests can be performed for each step. Thus, EXEC COND parameters are useful if the same return code has different meanings in different job steps, or if you want to take different actions according to which job step produced a return code.

  The system evaluates a COND parameter on the first EXEC statement in a job as false. However, you can use an IF statement before the first EXEC statement in a job to bypass the step.
- **The JOB COND parameter, when EXEC statements also contain COND parameters**, performs the same return code tests for every step in the job.
  - If the JOB statement return code test is satisfied, the job terminates. The job terminates regardless of whether or not any EXEC statements contain COND parameters and whether or not an EXEC return code test would be satisfied.
  - If the JOB statement return code test is not satisfied, the system then checks the COND parameter on the EXEC statement for the next step. If the EXEC statement return code test is satisfied, the system bypasses that step and begins processing of the following step, including return code testing.

  The COND parameter on both the JOB and EXEC statements is useful to set some conditions for all steps in the job and other conditions for particular steps.
- **No COND parameters on JOB or EXEC statements** means the system does not perform any return code tests, but tries to execute each step in the job.

Step Execution after a Preceding Step Abnormally Terminates

Abnormal termination of a step usually causes the system to bypass subsequent steps and to terminate the job. However, the EXEC statement COND parameter lets you request execution of a step by coding:

```
//stepname EXEC PGM=x,COND=EVEN
```

The step is to be executed even if one or more of the preceding steps abnormally terminates. That is, the step will always be executed, whether or not a preceding step abnormally terminates.

```
//stepname EXEC PGM=x,COND=ONLY
```

The step is to be executed only if one or more of the preceding steps abnormally terminates. That is, the step will not be executed, unless a preceding step abnormally terminates.

If a step abnormally terminates, the system scans the EXEC COND parameter for the next step for an EVEN or ONLY subparameter. If neither is present, the system bypasses the step. If EVEN or ONLY is specified, the system makes any requested return code tests against the return codes from previous steps that executed and

did not abnormally terminate. The step is bypassed if any test is satisfied. Otherwise, the step is executed.

Note: Certain error conditions prevent the system from executing a step, regardless of any requests specified through the COND parameter. Other considerations are also related to the use of the COND parameter. For information on cautions when specifying COND parameters, see the description of the COND parameter on the EXEC statement in *OS/390 MVS JCL Reference*.

Compatible Return Code Tests: The system applies the return code tests on the JOB COND parameter against the return code, if any, produced by each step in the job. To take advantage of this parameter, the return codes for each step should have compatible meanings. For example, the COBOL compiler and the linkage editor have compatible return codes:

4 Minor errors were found, but a compiled program or load module was produced. Execution may be successful.

8 Major errors were found, but a compiled program or load module was produced. Execution will probably not be successful.

12 Serious errors were found. A compiled program or load module was not produced.

Code the return code as follows:

COND=(4,LT) if you want to continue processing despite the minor errors. The job terminates only if the return code of any step is greater than 4.
COND=(4,LE) if you want to continue processing only if no errors occur. The job terminates if the return code of any step is greater than or equal to 4.

Examples of JOB Statement Return Code Tests

Example 1:

```
//J1  JOB  ,'LEE BURKET',COND=((10,GT),(20,LT))
```

This example asks ‘Is 10 greater than the return code or is 20 less than the return code?’. If either is true, the system skips all remaining job steps. If both are false after each step executes, the system executes all job steps.

For example, if a step returns a code of 12, neither test is satisfied. The next step is executed. However, if a step returns a code of 25, the first test is false, but the second test is satisfied: 20 is less than 25. The system bypasses all remaining job steps.

Example 2:

```
//J2  JOB  ,'D WEISKOPF',COND=((50,GE),(60,LT))
```

This example says ‘If 50 is greater than or equal to a return code, or 60 is less than a return code, bypass the remaining job steps.’ In other words, the job continues as long as the return codes are 51 through 60.

Example 3:

```
//J3  JOB  ,'E. SASSMANN',COND=(8,NE)
```

This example shows one return code test.

Example 4:

```
//J4 JOB  COND=((5,GT),(8,EQ),(12,EQ),(17,EQ),(19,EQ),(21,EQ),(23,LE))
```

This example shows seven return code tests. The job continues *only* if the return codes are: 5, 6, 7, 9, 10, 11, 13, 14, 15, 16, 18, 20, or 22.

Examples of EXEC Statement Return Code Tests

Example 1:

```
//S3 EXEC PGM=U,COND=((20,GT,STEP1),(60,EQ,STEP2))
```

This example says 'Bypass this step if 20 is greater than the return code STEP1 issues, or if STEP2 issues a return code of 60.'

Example 2:

```
//S4 EXEC PGM=V,COND=((20,GT,STEP1),(60,EQ))
```

This example says 'Bypass this step if 20 is greater than the return code STEP1 issues, or if any preceding step issues a return code of 60'.

Example 3:

```
//T7 EXEC PGM=B15,COND=(10,LT)
//STEP8 EXEC PGM=MYPROG,COND=(15,NE,STEP5)
```

These examples show single return code tests.

Example 4:

```
//NEXT EXEC PGM=AFTERPRC,COND=(7,LT,STEP4.LINK)
```

This example says 'Bypass this step if 7 is less than the return code issued by a procedure step named LINK in the cataloged procedure called by the EXEC statement named STEP4'.

Example 5:

```
//RCERROR EXEC PGM=ABEND,COND=(4,GE)
```

This example shows a single return code test. When you do not code a stepname, the step RCERROR will execute only when the return codes of all previous steps do not satisfy the test specified by COND.

Examples of EXEC COND Parameters with EVEN and ONLY

Example 1:

```
//S5  EXEC PGM=R,COND=EVEN
//R8  EXEC PGM=S,COND=((5,LT),EVEN)
//S6  EXEC PGM=T,COND=ONLY
//CX  EXEC PGM=U,COND=((4,GE,STEP3),(8,EQ,STEP2),ONLY,(12,LT,BX))
```

Example 2:

```
//LATE EXEC PGM=CLEANUP,COND=EVEN
```

This example says 'Execute program CLEANUP even if one or more of the preceding steps abnormally terminated.'

Example 3:

```
//LATER EXEC PGM=SCRUB,COND=((10,LT,STEPA),(20,EQ),ONLY)
```

This example says 'Execute this step only if one of the preceding steps terminated abnormally; but bypass it if 10 is less than the return code STEPA issues or if any of the steps that terminated normally issued a return code of 20'.

Example 4:

```
//LATEST EXEC PGM=FIX,COND=((10,LT,STEPA),(20,EQ),EVEN)
```

This example says 'Bypass this step if 10 is less than the return code STEPA issues, or if any of the preceding steps issues a return code of 20; otherwise execute this step even if one of the preceding steps terminated abnormally'.

Example 5:

```
//EXG EXEC PGM=A1,COND=(EVEN,(4,GT,STEP3))
//EXH EXEC PGM=A2,COND=((8,GE,STEP1),(16,GE),ONLY)
//EXI EXEC PGM=A3,COND=((15,GT,STEP4),EVEN,(30,EQ,STEP7))
```

Examples of COND Return Code Testing in a Job

Input Stream	*RC*	*Tests Performed*
`//MYJOB JOB  ,A.SMITH,COND=(10,LT)`		
`//STEP1 EXEC PGM=A`	6	Before STEP2: 1. Is 10 less than 6? No. 2. Is the return code 2 or 4? No. Execute STEP2
`//STEP2 EXEC PGM=B,COND=((2,EQ),(4,EQ))`	2	Before STEP3: 1. Is 10 less than 2 or 6? No. 2. Did one or more of the preceding steps terminate abnormally? No. Bypass STEP3.
`//STEP3 EXEC PGM=C,COND=ONLY`	–	Before STEP4: 1. Is 10 less than 2 or 6? No. 2. Is 5 greater than 6? No. 3. Is one of the preceding return codes equal to 2? Yes. Bypass STEP4.

`//STEP4 EXEC PGM=D,`
`//  COND=((5,GT,STEP1),(2,EQ))`
.
.
	–	Before STEP5:
		1. Is 10 less than 2 or 6? No.
		Execute STEP5.

`//STEP5 EXEC PGM=E`	9	Before STEP6: 1. Is 10 less than 9, 2, or 6? No. 2. Is 8 greater than 9? No. 3. Did one of the preceding steps terminate abnormally? No. Execute STEP6.

Input Stream	*RC*	*Tests Performed*
`//STEP6 EXEC PGM=F,`	10	Before STEP7:
`//  COND=((8,GT,STEP5),EVEN)`		1. Is 10 less than 10, 9, 2, or 6? No.
.		2. Is 4 greater than return code of STEP4?
.		STEP4 was bypassed and did not produce a
.		return code so this test evaluates as FALSE.
.		Execute STEP7.
.		
.		
`//STEP7 EXEC PGM=G,COND=(4,GT,STEP4)`	12	Before STEP8:
.		1. Is 10 less than 12, 10, 9, 2, or 6? Yes.
.		Bypass STEP8 and STEP9.
.		
`//STEP8 EXEC PGM=H`	–	
.		
.		
.		
`//STEP9 EXEC PGM=I,COND=ONLY`	–	
`//ABC JOB 12345,COND=(5,EQ)`		
`//STEP1 EXEC PGM=A`	4	Before STEP2:
.		1. Is 5 equal to 4? No.
.		2. Is 7 less than 4? No.
.		Execute STEP2.
.		
`//STEP2 EXEC PGM=B,COND=(7,LT)`	ABEND	Before STEP3:
.		1. Is EVEN or ONLY specified in
.		STEP3? Yes.
.		2. Is 5 equal to 4? No.
.		3. Is 20 greater than 4? Yes.
.		Bypass STEP3.
`//STEP3 EXEC PGM=C,`	–	Before STEP4:
`//  COND=((20,GT,STEP1),EVEN)`		1. Is EVEN or ONLY specified in
.		STEP4? Yes.
.		2. Is 5 equal to 4? No.
.		3. Are any preceding return codes
.		equal to 3? No. Execute STEP4.
.		
`//STEP4 EXEC PGM=D,COND=((3,EQ),ONLY)`	6	Before STEP5:
.		1. Is EVEN or ONLY specified in
.		STEP5? No. Bypass STEP5.
.		
`//STEP5 EXEC PGM=E,COND=(2,LT,STEP3)`	–	Before STEP6:
.		1. Is EVEN or ONLY specified in
.		STEP6? No. Bypass STEP6.
.		
`//STEP6 EXEC PGM=F`	–	Before STEP7:
.		1. Is EVEN or ONLY specified in
.		STEP7? Yes.
.		2. Is 5 equal to 6 or 4? No.
.		3. Is 6 equal to the return code of STEP5?
.		STEP5 was bypassed and did not produce a
.		return code so this test evaluates as FALSE.
.		Execute STEP7.
`//STEP7 EXEC PGM=G,`	5	Before STEP8:
`//  COND=((6,EQ,STEP5),ONLY)`		1. Is 5 equal to 5, 6, or 4? Yes.
.		Bypass STEP8 and STEP9.
.		
.		

Input Stream	*RC*	*Tests Performed*
`//STEP8 EXEC PGM=H,COND=EVEN`	–	
.		
.		
.		
`//STEP9 EXEC PGM=I`	–	

Examples of COND Parameters in Procedures

Example 1:

```
//TEST  EXEC  PROC=PROC4,COND.STEP4=((7,LT,STEP1),
//            (5,EQ),EVEN),COND.STEP6=((2,EQ),
//            (10,GT,STEP4))
```

In this example, the EXEC statement that calls procedure PROC4 passes COND parameters to two steps, STEP4 and STEP6,

Example 2:

```
//TEST  EXEC  PROC=MYPROC,COND=((7,LT,STEP1),(5,EQ))
```

This EXEC statement establishes a COND parameter for all steps in the called procedure. It overrides any COND parameters in the procedure, if coded.

Example 3:

```
//PS3  EXEC  PGM=ADD3,COND=(5,EQ,STEP2)
```

In this EXEC statement in a procedure, STEP2 in the COND parameter can be the name of either a preceding step in the procedure or of a preceding step in the job.

Example 4:

Your job contains	**Cataloged Procedure** PRA
. . `//TWO  EXEC PROC=PRA` . . .	. `//EDIT EXEC` . .
. . . `//THREE  EXEC PROC=PRB,COND.SP3=(10,LT,TWO.EDIT)`	**Cataloged Procedure** PRB . `//SP3 EXEC` . .

This example shows a procedure EXEC statement COND parameter that tests the return code from a step in another procedure called by a previous step in this job.

1. Step TWO calls cataloged procedure PRA, which contains procedure step EDIT. The system is to test the return code from EDIT.
2. Step THREE calls cataloged procedure PRB, which contains procedure step SP3. Execution of SP3 should depend on the return code from EDIT.
3. The COND parameter in EXEC statement THREE directs the system to bypass SP3 if 10 is less than the return code from procedure step EDIT.

The COND parameter could also have appeared on EXEC statement SP3:

```
//SP3 EXEC PGM=DEPEND,COND=(10,LT,TWO.EDIT)
```

To direct the system to bypass all steps in procedure PRB, code the COND parameter without the SP3 qualifier, as follows:

```
//THREE  EXEC  PRB,COND=(10,LT,TWO.EDIT)
```

Examples of COND Parameters that Force Step Execution

```
//S1   EXEC PGM=A
.
.
.

//CLEANUP EXEC PGM=FIX,COND=(12,NE,S1)
```

In this example, you force step CLEANUP to execute if step S1 executes but issues a return code of 12 to indicate that data sets might contain invalid records. The program FIX would clean up the invalid records.

Processing Control by Cancelling a Job that Exceeds Output Limit

You can control job execution by requesting cancellation of a job when its output exceeds a specified limit. The way you specify the limit depends on the environment in which your job is executing.

Limiting Output in an APPC Scheduling Environment

In an APPC scheduling environment, use the BYTES, CARDS, LINES, and PAGES parameters of the JOB statement to limit the number of:

- Bytes to be spooled for the job
- Cards to be punched for the job
- Lines to be printed for the job
- Pages to be printed for the job.

When you code the CANCEL subparameter with any of these parameters, the system cancels the job when the output exceeds the limit you have specified.

If you do not code a limit on the JOB statement BYTES, CARDS, LINES, or PAGES parameter, the system cancels the job when its output exceeds the installation default limit specified at JES initialization, and the JES cancel option has been specified.

Limiting Output in a Non-APPC Scheduling Environment

In a non-APPC scheduling environment, you can specify an output limit using the JOB statement parameters and installation defaults described in Limiting Output in an APPC Scheduling Environment. In addition, you can code a BYTES, CARDS, LINES, or PAGES parameter on a JES2 /*JOBPARM statement or a JES3 //*MAIN statement. These parameters limit the number of:

- Bytes to be spooled for the job
- Cards to be punched for the job
- Lines to be printed for the job
- Pages to be printed for the job.

When you code the CANCEL subparameter on the //*MAIN statement, the system cancels the job when its output exceeds the limit you have specified.

When you code an output limit on the /*JOBPARM statement, the system cancels the job when:

- The job's output exceeds the limit you have specified, and
- The cancel option has been specified at JES2 initialization as the installation default.

If you do not code an output limit on the JOB statement, the system uses the limit coded on the //*MAIN statement or the /*JOBPARM statement. If you do not code a //*MAIN or a /*JOBPARM statement, the system uses the installation default limit specified at JES initialization, and cancels the job if the JES cancel option has been specified.

Use in Testing

One use for the output limit is during program testing. You can cancel a program that is in an endless loop containing instructions that send records to a sysout data set.

Examples:

The following examples illustrate the use of the JCL JOB statement, in either an APPC or non-APPC scheduling environment, to warn the operator when the output for a job has exceeded a limit in any JES system:

```
//JOB1  JOB  ACCT01,'D. PIKE',BYTES=(50,CANCEL)

//JOB2  JOB  1542,RWALLIN,CARDS=(120,CANCEL)

//JOB3  JOB  ,ZOBES,LINES=(200,CANCEL)

//JOB4  JOB  ACCT27,'S M SHAY',PAGES=(,CANCEL)
```

The following examples illustrate the use of the JES3 //*MAIN statement in a non-APPC scheduling environment to warn the operator when output for a job has exceeded a limit.

```
//*MAIN  BYTES=(50,CANCEL)
//*MAIN  CARDS=(120,CANCEL)
//*MAIN  LINES=(200,CANCEL)
//*MAIN  PAGES=(,CANCEL)
```

Processing Control by Timing Execution

To control processing based on the processor time needed to execute a program, code one of the following time parameters:

```
//jobname  JOB  acct,progname,TIME=value
//stepname EXEC PGM=x,TIME=value
//jobname  JOB  (,,time)
/*JOBPARM  TIME=value
```

JOB and EXEC TIME Parameter

The TIME parameter on the JOB or EXEC statement specifies the maximum length of time a job or step is to use the processor. Two benefits of the TIME parameter are:

- The system prints the actual processor time used by the job or step in the messages in the job log.
- When a job or step exceeds the amount of time coded on the TIME parameter, the system abnormally terminates it or gives control to an installation exit routine established through System Management Facilities (SMF). Thus, the TIME value limits the processor time wasted by a looping program.

By coding TIME=1440 or TIME=NOLIMIT, the TIME parameter can instead be used to give a job or step an unlimited amount of time. Specifically, the system allows a step to remain in a continuous wait state for an unlimited time, rather than the time limit established through SMF. However, if TIME=1440 is specified on the JOB statement, any TIME values on an EXEC statement and any default TIME values will be nullified. All steps within the job will have unlimited time, as with TIME=1440 or TIME=NOLIMIT.

To allow a job or step to use the maximum amount of time, code TIME=MAXIMUM. Coding TIME=maximum allows the job or step to run for 357912 minutes.

Example 1:

```
//FIRST JOB   ,'E.D. WILLIAMSON',TIME=2
//STEP1 EXEC  PGM=A,TIME=1
//STEP2 EXEC  PGM=B,TIME=1
```

In this example, the job is allowed 2 minutes of execution time and each step is allowed 1 minute. Should either step try to execute beyond 1 minute, the job will terminate beginning with that step.

Example 2:

```
//SECOND JOB   ,'M. CARLO',TIME=3
//STEP1  EXEC  PGM=C,TIME=2
//STEP2  EXEC  PGM=D,TIME=2
```

In this example, the job is allowed 3 minutes of execution time. Each step is allowed 2 minutes of execution time. Should either step try to execute beyond 2 minutes, the job will terminate beginning with that step. If STEP1 executes in 1.74 minutes and if STEP2 tries to execute beyond 1.26 minutes, the job will be terminated because of the 3-minute time limit specified on the JOB statement.

Example 3:

```
//THIRD JOB   ,'A. DOMENICK',TIME=2
//STEP1  EXEC  PGM=E,TIME=3
```

In this example, the job is allowed 2 minutes of execution time. Since the time specified on the JOB statement is less than the time on the EXEC statement, STEP1 is only allowed 2 minutes of execution time. If STEP1 attempts to execute beyond 2 minutes, the job will terminate in that step.

Example 4:

```
//AAA  EXEC  PROC=PROC5,TIME=20
```

In this example, the EXEC statement sets a time limit for an entire procedure. This specification overrides any TIME parameters in the procedure, if coded.

Example 5:

```
//AAA  EXEC  PROC=PROC5,TIME.ABC=20,TIME.DEF=(3,40)
```

In this example, the EXEC statement sets a time limit for two steps, ABC and DEF, of the called cataloged procedure.

JES2 Time Parameters

In a JES2 system, you can code a time value in the JES2 format accounting information parameter on the JOB statement or in a TIME parameter on the JES2 /*JOBPARM statement. If the job execution time exceeds this value, JES2 sends a message to the operator.

Examples:

```
//J3  JOB  (,,3)
/*JOBPARM  TIME=3
```

Both of these statements specify that the job cannot use the processor for more than 3 minutes.

OS/390 UNIX System Services Considerations

In OS/390 UNIX System Services, callable service BPX1SRL lets a program modify its job time. See *OS/390 UNIX System Services Programming: Assembler Callable Services Reference* for more information on the BPX1SRL callable service.

Processing Control for Testing

You can test your JCL for errors by using one of the following methods.

Altering Usual Processing for Testing

These testing methods change the standard job processing to allow the system to find errors.

Scanning JCL for Errors (Non-APPC)

The TYPRUN and CLASS parameters described in this section have no effect in an APPC scheduling environment. If you code them, the system will check them for syntax and ignore them.

Before using a new set of job control statements, you can ask the system to scan them for syntax errors without executing any steps or allocating any devices. To do this scanning, code:

- For a job in a JES2 or JES3 system:

```
//jobname  JOB  acct,progname,TYPRUN=SCAN
```

- For a job in a JES2 system, where x is a class defined during JES2 initialization to force job control statement scanning:

```
//jobname  JOB  acct,progname,CLASS=x
```

- For a step in a JES3 system:

```
//stepname EXEC PGM=JCLTEST
//stepname EXEC PGM=JSTTEST
```

The system scans for:

- Invalid spelling of parameter keywords and some subparameter keywords.
- Invalid characters.
- Unbalanced parentheses.
- Misplaced positional parameters on some statements.
- In a JES3 system only, parameter value errors or excessive parameters.
- Invalid syntax on JCL statements in cataloged procedures invoked by any scanned EXEC statements.

The system does not check for misplaced statements, for invalid syntax in JCL subparameters, or for parameters and/or subparameters that are inappropriate together.

Examples:

```
//JB16 JOB   ,'M. CARLO',TYPRUN=SCAN
//TG   JOB   RK988,SMITH,CLASS=S
//S1   EXEC  PGM=JCLTEST
//S2   EXEC  PGM=JSTTEST
```

Using IEFBR14 Program for Testing

IEFBR14 is a two-line program that clears register 15, thus passing a return code of 0, and then branches to the address in register 14, which returns control to the system. If a step requests IEFBR14 instead of the program that the JCL actually supports, the system does the following:

- Checks all the job control statements in the step for syntax.
- Allocates direct access space for data sets.
- Performs data set dispositions.

To test with IEFBR14, substitute IEFBR14 for the name of the program, as follows:

```
//stepname EXEC PGM=IEFBR14,...
```

Considerations when Using IEFBR14: Although the system allocates space for data sets, it does not initialize the data sets. Therefore, any attempt to read from one of these data sets will produce unpredictable results. Also, IBM does not recommend allocation of multi-volume data sets while executing IEFBR14.

If you created a data set when testing with IEFBR14, the data set's status in the DD DISP parameter is old when you execute the actual program.

Because IEFBR14 does not open any data sets, a DD DISP parameter of CATLG does not make the system catalog a data set, if one of the following is true:

- The DD statement requested a nonspecific tape volume.
- The DD statement requested a tape volume with dual density options, but the DCB DEN subparameter did not specify the density.
- The DD statement was allocated to a tape volume with dual recording mode options, but you did not code the DCB TRTCH subparameter.

When executing IEFBR14, if a DD DISP parameter specifies CATLG or UNCATLG, the system issues an operator message to mount the volume. If it is not necessary to mount the volume, code DEFER on the UNIT parameter of the DD statement.

Examples:

```
For testing:
//STEP1  EXEC  PGM=IEFBR14,COND=(8,LE),TIME=2

For executing after testing:
//STEP1  EXEC  PGM=WKLYRPT,COND=(8,LE),TIME=2
```

Using Nonstandard Processing

In a JES3 system, you can use nonstandard job processing in testing. Standard job processing consists of the following standard scheduler functions:

Converter/interpreter service
Main service
Output service
Purge service

A nonstandard job uses one or more special processing functions in place of or in addition to the standard functions or skips one or more standard functions. Specify nonstandard processing by following the JOB statement with a JES3 //*PROCESS statement for each processing function to be performed.

End the //*PROCESS statements with a //*ENDPROCESS statement or a JCL statement.

Example:

```
//TESTA    JOB  ,'E. HARMANTAS'
//*PROCESS CI
//STEP1    EXEC PGM=NEWPROG
//DD28     DD   SYSOUT=A
//DD29     DD   *
           .
           .
          (data)
           .
/*
```

This example asks for only the converter/interpreter service, CI. The converter/interpreter scans the job's syntax for errors. The program will not be executed or the job's output processed. However, the job will be purged from the system.

Dumping after Error

To request that the system dump the storage occupied by a failing program and other storage needed to debug the program, code one of the following:

- SYSABEND, SYSMDUMP, or SYSUDUMP DD statement in the step to be dumped. The system produces the requested dump if the step terminates abnormally or if the step starts to terminate abnormally, but the system recovery procedures allow the step to terminate normally.

  If there are multiple failures in the same job step, only the last failure is reported. Therefore, inspect the dump to gather information about any possible earlier failures.

- DUMP in the BYTES, CARDS, LINES, or PAGES parameter of the JOB statement. The system produces the dump requested by the dump DD statement for the step if the system cancels the job because:

  1. The job's output exceeds the maximum specified on the BYTES, CARDS, LINES, or PAGES parameter of the JOB statement, or
  2. The job's output exceeds the maximum output specified on the JES3 //*MAIN statement, or the JES2 /*JOBPARM statement
  3. The job's output exceeds the maximum defined by the installation defaults specified at initialization.

- DUMP in the BYTES, CARDS, LINES, or PAGES parameter on the JES3 //*MAIN statement in the job and a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement in the step to be dumped. The system produces the dump requested by the dump DD statement if JES3 cancels the job because the job's output exceeds the BYTES, CARDS, LINES, or PAGES limit or, if no limits are given, the installation default limit for the job class.

If the dump is to be printed directly on a 3800 Printing Subsystem, the SYSABEND or SYSUDUMP DD statement can request a high-density dump by specifying:

- FCB=STD3 to produce dump output at 8 lines per inch.
- CHARS=DUMP to produce 204-character print lines.

Example 1:

```
//S1       EXEC PGM=TESTING
//DS1      DD   SYSOUT=C
//SYSABEND DD   SYSOUT=A,FCB=STD3,CHARS=DUMP
//INDS     DD   *
              .
              .
           (data)
              .
/*
```

This example produces a high-density dump, if TESTING abnormally terminates.

Example 2:

```
//J3JB     JOB  ,'J.T. HIGGINS',MSGCLASS=B
//*MAIN    LINES=(50,DUMP)
//S1       EXEC PGM=OLDPROG
              .
              .
              .
//S2       EXEC PGM=NEWPROG
//SYSUDUMP DD   SYSOUT=D
              .
              .
              .
```

If the first step exceeds 50,000 lines of output, JES3 cancels the job but does not write a dump because the first step does not contain a dump DD statement. If the combined output from S1 and S2 exceeds 50,000 lines, JES3 cancels the job and writes a SYSUDUMP dump to the sysout data set for class D.

Example 3:

```
//JOB1     JOB  ,'W. BAILEY',MSGCLASS=B,BYTES=(30,DUMP)
//STEP1    EXEC PGM=TESTPGM
//SYSUDUMP DD   SYSOUT=D
              .
              .
              .
```

If the first step exceeds 30,000 lines of output, the system cancels the job and writes a SYSUDUMP dump to the sysout data set for class D.

Chapter 11. Processing Jobs - Performance Control

The performance control described in this chapter is not supported in an APPC scheduling environment, with the exception of the performance control described in “Performance Control by Performance Group (Non-APPC)” on page 11-3.

Figure 11-1. Performance Control Task for Processing Jobs

TASKS FOR PROCESSING JOBS	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	JOB	EXEC	Other JCL		
Performance Control					
by job class assignment	CLASS				CLASS on //*MAIN
by selection priority	PRTY			/*PRIORITY	
by performance group assignment	PERFORM	PERFORM			
by I/O-to-processing ratio					IORATE on //*MAIN

Performance Control by Job Class Assignment (Non-APPC)

The performance control described in this topic is not supported in an APPC scheduling environment.

The system can balance the mix of jobs being executed based on the class and priority assigned to each job. The installation should assign classes and priorities so that the system will not simultaneously execute jobs that compete for the same resources.

A JES2 installation can have 36 job classes; a JES3 installation can have 255 job classes. Two more classes are reserved for started tasks and time sharing users. The installation arbitrarily determines the type of job to be placed in each class. In general, all jobs with the same characteristics should be in the same class.

For example, an installation can identify separate classes for the following job types:

- I/O-bound jobs.
- Processor-bound jobs.
- Jobs being debugged.
- Jobs using a particular resource.

Using these example job classes, the installation can assign job classes so that:

- I/O-bound jobs will execute at the same time as processor-bound jobs. This multiprogramming helps both types of jobs complete faster.
- All programs that use tape drives will be in the same class, if the installation contains only a few tape drives.

© Copyright IBM Corp. 1988, 1999

- All programs that use a data base will be in the same class, if the data base must be accessed serially.

The installation should maintain a list of job classes and the types of jobs to be assigned to them.

In a JES2 system, assign a job to a job class by coding:

```
//jobname JOB acct,progname,CLASS=x
```

In a JES3 system, assign a job to a job class, which is part of a job class group, by coding either of the following:

```
//jobname JOB acct,progname,CLASS=x
//*MAIN CLASS=x
```

Examples

```
//MYJOB  JOB  ACCT24,BIRDSALL,CLASS=F
//*MAIN  CLASS=H
```

Performance Control by Selection Priority (Non-APPC)

The performance control described in this topic is not supported in an APPC scheduling environment.

Within a JES2 job class or a JES3 job class group, the system selects jobs for execution in order by priority. The higher the priority number, the sooner the job is selected. Jobs with the same priority are selected on a first-in first-out basis.

Priority for JES2 Jobs

In a JES2 system, there are a number of factors that determine the order in which a particular job is selected for execution. Therefore, you cannot be assured that job priority (based on the PRTY you assign a job) or the order of job submission will guarantee that the jobs will execute in a particular order. If you need to submit jobs in a specific order, contact your JES2 system programmer for advice based on how your system honors such requests. (*OS/390 JES2 Initialization and Tuning Guide* provides JES2 system programmer procedures concerning job queuing and how to control job execution sequence.)

If a priority is not specified, JES2 uses installation algorithms to calculate the job's priority based on the execution time and the estimated amount of output. The operator can assign a different priority or you can code one of the following:

```
//jobname JOB acct,progname,PRTY=x
/*PRIORITY x
```

JES2 also uses the execution time and output amount to monitor job execution time and output. If you do not code these estimates, JES2 assumes installation defaults. If your job exceeds the coded or assumed estimates, JES2 issues warning messages to the operator or cancels the job, with or without a dump.

Use of Priority

An installation can specify that jobs with shorter execution times and less output should be assigned higher priorities. To make sure that programmers specify

correct times and output, the installation can instruct the operator to cancel jobs that exceed the estimates.

Examples

```
//JOB10  JOB  ,'FLO JONES',PRTY=14
/*PRIORITY  14
```

Priority for JES3 Jobs

To assign a priority to your job, you can code the following:

```
//jobname JOB acct,progname,PRTY=x
```

The operator can change a job's priority; see *OS/390 JES3 Commands*.

Example

```
//JOB10  JOB  ,'FLO JONES',PRTY=14
```

Priority Aging

JES2 increases the priority of a job as it waits to be executed in the system. JES2 keeps raising the job's priority until it is executed.

JES3 increases a job's priority based on the number of times the job is passed over for selection. A job can be passed over because not enough devices are available or because another job has a needed volume or data set or because not enough storage is available.

The installation defines priority aging; you cannot specify it using JCL.

Performance Control by Performance Group (Non-APPC)

The performance control described in this topic is not supported in workload management goal mode.

Performance groups determine how fast a job executes by controlling the rate at which jobs in the group have access to the processor, the main storage, and the I/O channels. The installation defines the performance groups. Most performance groups designate good processing rates under light system workloads. However, when the system workload is moderate or heavy, some performance groups have much lower processing rates than others.

The installation should define performance groups to meet the response requirements of the jobs to be executed. The installation should maintain a list of these groups.

To associate a job or job step with a performance group, code:

```
//jobname  JOB  acct,progname,PERFORM=n
//stepname EXEC PGM=x,PERFORM=n
```

Note: The PERFORM parameter regulates how a job executes as contrasted with the JES3 //*MAIN IORATE parameter, which regulates how a job is scheduled.

For more information on performance, see *OS/390 MVS Initialization and Tuning Guide* and *OS/390 JES2 Initialization and Tuning Guide* or *OS/390 JES3 Initialization and Tuning Guide*.

Examples

```
//J71   JOB  ,'ANTHONY B.',PERFORM=52
//STEPC EXEC PGM=WHIT,PERFORM=4
```

Performance Control by I/O-to-Processing Ratio (Non-APPC)

The performance control described in this section is not supported in an APPC scheduling environment.

To regulate how a job is scheduled by JES3, code an IORATE parameter:

```
   //*MAIN IORATE=xxx
```

The IORATE parameter indicates if the job contains a low, medium, or high number of I/O instructions compared to the number of processing instructions. JES3 uses this value to determine the mix of jobs assigned to a processor: using this parameter, JES3 balances processor-bound processing with I/O-bound processing. A correct balance improves throughput.

Examples

```
//*MAIN IORATE=HIGH
//*MAIN IORATE=LOW
//*MAIN IORATE=MED
```

Part 4. Tasks for Requesting Data Set Resources

Chapter 12. Data Set Resources - Identification 12-1
Identification of Data Set 12-1
Permanent Data Set 12-1
Members of a PDS or PDSE 12-2
Generations of a Generation Data Group 12-2
Areas of an Indexed Sequential Data Set 12-3
Temporary Data Sets 12-3
Members of a Temporary PDS or PDSE 12-4
Areas of a Temporary Indexed Sequential Data Set 12-4
Copying the Data Set Name from an Earlier DD Statement 12-5
Concatenating Data Sets 12-5
Identification of In-Stream Data Set (Non-APPC) 12-5
Entering Data Through the Input Stream 12-5
In-Stream Data Sets in a JES3 System 12-6
Identification of Data Set on 3540 Diskette Input/Output Unit 12-7
Identification through Catalog 12-7
Using Private Catalogs 12-8
Identification through Label 12-8
Identification by Location on Tape 12-10
Identification as TCAM Message Data Set 12-10
Identification as Data Set from or to Terminal (Non-APPC) 12-11

Chapter 13. Data Set Resources - Description 13-1
Description of Status 13-1
Data Set Integrity Processing 13-2
Summary of Data Set Integrity Processing 13-4
Description of Data Attributes 13-5
In Data Control Block (DCB) 13-5
In Access Method Control Block (ACB) 13-6
In Data Class 13-6
From Model Data Set 13-7
Migration and Backup (with SMS) 13-7

Chapter 14. Data Set Resources - Protection 14-1
Protection through RACF 14-1
Protection with the PROTECT Parameter 14-1
Protection with the SECMODEL Parameter 14-2
Protection for ISO/ANSI/FIPS Version 3 Tapes 14-2
Protection by Passwords 14-3
Protection of Access to BSAM or BDAM Data Sets 14-3

Chapter 15. Data Set Resources - Allocation 15-1
Allocation of Device 15-2
Device Allocation for SMS-Managed Data Sets 15-2
Device Allocation for Non-SMS-Managed Data Sets 15-3
Specifying Device Number 15-4
Specifying Device for Output Data Set (Non-SMS-Managed Data Sets) 15-5
Allocation with Deferred Volume Mounting 15-6
Requesting More than One Unit for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume 15-6

© Copyright IBM Corp. 1988, 1999

Number of Devices Allocated for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-6
Volumes Required per DD Statement . . . 15-6
Devices Required per DD Statement . . . 15-6
Devices Assigned per Step . . . 15-7
Examples for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-8
Device Allocation in a JES3 System . . . 15-12
Types of JES3 Setup . . . 15-13
Allocation of Volume . . . 15-16
Volume Allocation for SMS-Managed Data Sets . . . 15-17
References to SMS-Managed Data Sets . . . 15-17
Specific Volume Requests for System-Managed DASD Data Sets . . . 15-17
Nonspecific Volume Requests for System-Managed Data Sets . . . 15-18
Multivolume Data Sets for System-Managed DASD Data Sets . . . 15-18
Volume Allocation for Non-SMS-Managed Data Sets . . . 15-18
Volume Allocation for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-18
Specific Volume Requests for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-19
Nonspecific Volume Requests for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-20
Private Volumes for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-21
Public Volumes for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-21
Volume Affinity for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-21
Multivolume Data Sets for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-22
Volumes Required per DD Statement for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-23
Interactions Between Device and Volume Allocation . . . 15-24
Relationship of the UNIT and VOLUME Parameters (Non-SMS-Managed Data Sets) . . . 15-24
Relationship of the UNIT and VOLUME Parameters (SMS-Managed Data Sets) . . . 15-29
Unit and Volume Affinity for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume . . . 15-30
Interaction of Unit and Volume Affinity Requests . . . 15-34
Affinity for Multivolume Data Sets . . . 15-36
Device Usage for Data Sets on a System-Managed Tape Volume . . . 15-36
Examples of When the System Ignores Unit Affinity . . . 15-36
Stacking Data Sets . . . 15-39
Examples of Data Set Stacking . . . 15-40
Data Set Stacking and TMM . . . 15-42
Allocation of Direct Access Space . . . 15-44
Requesting System Assigned Space . . . 15-44
How the System Satisfies the Primary Space Request . . . 15-45
How the System Satisfies the Secondary Space Request . . . 15-46
Directory Space for Partitioned Data Sets . . . 15-47
System Assigned Space Requests for Indexed Sequential Data Sets 15-48
Requesting Specific Tracks . . . 15-49
Specific Track Requests for Indexed Sequential Data Sets . . . 15-49

Allocation of Virtual I/O 15-49
Backward References to VIO Data Sets 15-51
Allocation with Volume Premounting in a JES2 System 15-52
Dynamic Allocation 15-53

Chapter 16. Data Set Resources - Processing Control 16-1
Processing Control by Suppressing Processing 16-1
Processing Control by Postponing Specification 16-2
Processing Control with Checkpointing 16-4
Processing Control by Subsystem 16-4
Requesting Subsystem 16-4
Program Control Statements for a Subsystem 16-5
Processing Control by TCAM Job or Task 16-5

Chapter 17. Data Set Resources - End Processing 17-1
Unallocation End Processing 17-1
Disposition End Processing of Data Set 17-1
Disposition Controlled by DISP Parameter 17-2
Deleting a Data Set 17-3
Keeping a Data Set 17-3
Cataloging a Data Set 17-4
Uncataloging a Data Set 17-5
Passing a Data Set 17-5
Default Disposition Processing 17-9
Bypassing Disposition Processing 17-9
Disposition Processing of Data Sets that Do Not Exist 17-9
Disposition Controlled by Time 17-11
Release of Unused Direct Access Space in End Processing 17-12
Disposition End Processing of Volume 17-12
Disposition of Removable Volumes 17-13
Volume Retention 17-13

This part describes how to create and access data sets. The task required to request a data set is:

- Identification

Other tasks can optionally be performed:

- Description
- Protection
- Allocation
- Processing control
- End processing

Part 4. Tasks for Requesting Data Set Resources

Chapter 12. Data Set Resources - Identification

Figure 12-1. Identification Task for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Identification					
of data set	DSNAME				UPDATE on //*MAIN
of in-stream data set	* or DATA SYSIN DD DLM		 /* or xx delimiter		//*DATASET //*ENDDATASET
of data set on 3540 Diskette Input/Output Unit	DSID				
through catalog	JOBCAT DD STEPCAT DD				
through label	label-type on LABEL				
by location on tape	data-set-sequence-number on LABEL				
as TCAM message data set	QNAME				
from or to terminal	TERM				

Identification of Data Set

When creating a data set, assign a name to the data set in the DSNAME parameter. The data set name is stored with the data set. When a later step or job uses the data set, identify the data set in the DSNAME parameter; the system uses the data set name to locate the data set on the volume.

How you code the DSNAME parameter depends on the type of data set and whether it is permanent or temporary or it is copied from an earlier DD statement.

For information on allocation of data sets, refer to Chapter 15, “Data Set Resources - Allocation” on page 15-1.

Permanent Data Set

Identify a permanent data set by coding:

DSNAME=dsname	For a permanent data set
DSNAME=dsname(member)	For a member of a permanent PDS or PDSE
DSNAME=dsname(generation)	For a generation of a permanent generation data group

© Copyright IBM Corp. 1988, 1999

DSNAME=dsname(area)	For an area of a permanent indexed sequential data set

To create a permanent data set, assign it a name in the DSNAME parameter and a disposition of KEEP or CATLG in the DISP parameter. The DISP subparameter makes it a permanent data set. To use the data set, specify the data set's name in the DSNAME parameter in a later step or job or a backward reference to the creating DD statement in a later step in the same job.

Examples

```
//MYDS   DD  DSNAME=PLANA,DISP=(NEW,KEEP,DELETE),
//       UNIT=3380,VOLUME=SER=167833,SPACE=(CYL,(10,5))

//DSC    DD  DSNAME=PLANB,DISP=(NEW,CATLG,DELETE),
//       UNIT=3350,VOLUME=SER=275566,SPACE=(TRK,(20,5))

//SMSDS  DD  DSNAME=DESIGNB.PGM,DATACLAS=DCLAS1,STORCLAS=SCLAS1,
//       DISP=(NEW,KEEP)

//OLDDS  DD  DSNAME=EXIST,DISP=OLD
```

Members of a PDS or PDSE

A partitioned data set (PDS) and a partitioned data set extended (PDSE) consist of sequential records in independent groups, which are called members; each member is identified by a member name. To add a member to a PDS or a PDSE, or to retrieve a member, specify the data set name followed by the member name in parentheses.

Example (PDS)

```
//NEWA   DD  DSNAME=RPRT(WEEK1),DISP=(NEW,CATLG,DELETE),
//       UNIT=3380,VOLUME=SER=236688,SPACE=(CYL,(20,5,20))

//ADD1   DD  DSNAME=RPRT(WEEK2),DISP=OLD
```

Example (PDSE)

```
//SMSDS  DD  DSNAME=RPRT(WEEK1),DATACLAS=DCLAS1,STORCLAS=SCLAS1,
//       DISP=(NEW,KEEP)

//ADDSMS DD  DSNAME=RPRT(WEEK2),DISP=OLD
```

Generations of a Generation Data Group

A generation data group is a collection of chronologically related data sets that have the same data set name. To add a generation to a generation data group or retrieve a generation, specify the generation data group name followed by the generation number. A zero is the current generation of the group, a negative number (for example, -1) is an older generation, a positive number (for example, +1) is a new generation that does not exist yet.

Examples

```
//NEWGDS  DD  DSNAME=GDS(0),DISP=(NEW,CATLG,DELETE),
//        UNIT=3380,VOLUME=SER=334455,SPACE=(CYL,20)

//OLDGDS  DD  DSNAME=GDS(-1),DISP=OLD

//NEWER   DD  DSNAME=GDS(+1),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(TRK,15)

//ALLG    DD  DSNAME=GDS,DISP=OLD

//SMSGDG  DD  DSNAME=A.B.C(+1),DATACLAS=DGDG1,DISP=(NEW,KEEP)
```

Areas of an Indexed Sequential Data Set

An indexed sequential data set consists of three areas: index, prime, and overflow. To create the data set, define each area by identifying the data set name followed by the area name. The area name is INDEX, PRIME, or OVFLOW. To define the data set on one DD statement, code DSNAME=dsname or DSNAME=dsname(PRIME). To retrieve the data set, code only the data set name.

Examples

```
//NEWIS  DD  DSNAME=ISDS(INDEX),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(CYL,5)
//       DD  DSNAME=ISDS(PRIME),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(CYL,15)
//       DD  DSNAME=ISDS(OVFLOW),DISP=(NEW,CATLG,DELETE),
//        UNIT=3350,VOLUME=SER=222333,SPACE=(CYL,10)

//OLDIS  DD  DSNAME=ISDS,DISP=OLD
```

Temporary Data Sets

A temporary data set is a data set that is created and deleted in the same job, and is identified by coding one of the following:

DSNAME=&&dsname	For a temporary data set
DSNAME=&&dsname(member)	For a member of a temporary PDS or PDSE
DSNAME=&&dsname(area)	For an area of a temporary indexed sequential data set
No DSNAME parameter	For a temporary data set to be named by the system

Additionally, in a non-SMS environment only, the system treats any data set that is created and deleted in the same job step as a temporary data set. For example, the system treats a data set coded as:

```
DSN=A.REAL.DSN.NAME,DISP=(NEW,DELETE)
```

in a non-SMS environment as a temporary data set.

Only the job that creates a temporary data set has access to it to read and write data and to scratch the data set.

SMS manages a temporary data set if (1) you specify a storage class (via the DD STORCLAS parameter) or (2) an installation-written automatic class selection (ACS) routine selects a storage class for the temporary data set.

The system generates a qualified name for the temporary data set. For details about the format of the name the system generates, see the description of the DSNAME parameter in *OS/390 MVS JCL Reference*.

The time in the system-generated qualified name is the same for all temporary data sets in a job. Therefore, if the same temporary data set name appears more than once in a job, the system might create duplicate data set names. This would be a JCL error, unless the data set is passed from one job step to another.

If the DISP parameter for a temporary data set specifies KEEP or CATLG, the system changes the disposition to PASS and deletes the data set at job termination. However, the system does not change the disposition for a data set when all of the following are true:

- The data set resides on tape
- The data set is new
- The data set is not named in a DSNAME parameter
- The status in the DISP parameter is OLD or SHR
- The UNIT parameter contains DEFER

In this case, the system deletes the data set at job termination but tells the operator to keep the volume for the data set.

Examples

```
//TEMPDS1  DD  DSNAME=&&MYDS,DISP=NEW,UNIT=3350,
//         SPACE=(CYL,20)

//TEMPDS2  DD  DSNAME=&&DSA,DISP=(NEW,PASS),UNIT=3380,
//         SPACE=(TRK,15)

//TEMPSMS  DD  DSNAME=&&ABC,DATACLAS=DCLAS2,STORCLAS=TEMP1,DISP=NEW
```

Members of a Temporary PDS or PDSE

To add a member to a temporary partitioned data set (PDS or PDSE), or to retrieve a member during the job, specify the data set's temporary name and follow it with the member name in parentheses.

Examples

```
//TEMPMEM  DD  DSNAME=&&DS1(MEM1),DISP=(NEW,PASS),
//        UNIT=3380,SPACE=(CYL,(20,,2))

//GETMEM   DD  DSNAME=&&DS1(MEM1),DISP=OLD
```

Areas of a Temporary Indexed Sequential Data Set

To create a temporary indexed sequential data set and define any of its areas on a DD statement, identify the data set's temporary name followed by the area name. To define the temporary data set on one DD statement, code DSNAME=&&dsname or DSNAME=&&dsname(PRIME). To retrieve the temporary data set in the same job, code DSNAME=&&dsname.

Examples

```
//TEMPIS   DD  DSNAME=&&ISDS(INDEX),DISP=(NEW,PASS),
//         UNIT=3380,SPACE=(CYL,5)
//         DD  DSNAME=&&ISDS(PRIME),DISP=(NEW,PASS),
//         UNIT=3380,SPACE=(CYL,20)
//         DD  DSNAME=&&ISDS(OVFLOW),DISP=(NEW,PASS),
//         UNIT=3380,SPACE=(CYL,10)

//ANOTHER  DD  DSNAME=&&ISDS2,DISP=(NEW,PASS),UNIT=3350,
//         SPACE=(CYL,10)

//OLDIS    DD  DSNAME=&&ISDS2,DISP=OLD
```

Copying the Data Set Name from an Earlier DD Statement

If a data set name is used several times in a job, copy it from the DD statement that uses it first. It can be copied whether it is specified in the DSNAME parameter or assigned by the system. Use copying to make changing data sets from job to job easier and to eliminate having to assign names to temporary data sets. Copy a data set name by coding:

```
//ddname DD  DSNAME=*.ddname
//ddname DD  DSNAME=*.stepname.ddname
//ddname DD  DSNAME=*.stepname.procstepname.ddname
```

Example

```
//COPYDS  DD  DSNAME=*.MYDS
```

Concatenating Data Sets

You can logically connect or concatenate (link together) sequential or partitioned data sets (PDSs or PDSEs) for the duration of a job step. To concatenate data sets, omit the ddnames from all the DD statements except the first. The data sets are processed in the same sequence as the DD statements defining them.

Example

```
//INPUT  DD  DSNAME=FGLIB,DISP=(OLD,PASS)
//       DD  DSNAME=GROUP2,DISP=SHR
```

Identification of In-Stream Data Set (Non-APPC)

In-stream data sets are not supported in an APPC scheduling environment. If coded, the system will syntax-check and ignore the DD statement that identifies the in-stream data set. Subsequent statements will be processed as JCL statements and might cause errors. The system ignores a delimiter statement that follows the in-stream data set.

Entering Data Through the Input Stream

Enter data through the input stream by coding one of the following:

```
//ddname  DD  *
//ddname  DD  DATA
```

A step can contain more than one in-stream data set. Use the DD DATA statement when the data contains JCL statements.

If the statement that begins the data set contains a DLM parameter, end the in-stream data set with a statement containing the two characters in the DLM parameter. Otherwise, end the in-stream data set with either of the following delimiters:

```
/*
Another JCL statement, if begun with a DD * statement
```

Naming an In-Stream Data Set

Code the DSNAME parameter on the DD * or DATA statement to assign the last qualifier of the system-generated name to an in-stream data set.

Example 1

```
//DSIN  DD  *
        .
        .
        (data)
        .
//INSET DD  DATA
        .
        .
        (data)
        .
/*
//THIRD DD *,DLM=ED
        .
        .
        (data)
        .
ED
```

Example 2

```
//DDIN  DD  DATA,DSNAME=&&PAYIN1
        .
        .
        (data)
        .
/*
```

In-Stream Data Sets in a JES3 System

In a JES3 system, an in-stream data set can also begin with a //*DATASET statement and end with a //*ENDDATASET statement. The //*DATASET statement must start an in-stream data set that is used as input to a dynamic support program (DSP).

Example

```
//J1   JOB   2233,'K.A. BRAND'
//S1   EXEC  PGM=MYPROG
//*DATASET DDNAME=S1.MYDD4,J=YES
        .
        .
       data
        .
//*ENDDATASET
```

Identification of Data Set on 3540 Diskette Input/Output Unit

IBM 3540 diskette volumes can contain associated data sets. Associated data sets are treated like in-stream data sets and are spooled in as SYSIN data sets. These associated data sets are identified by coding a DSID parameter and, optionally, a volume serial on a DD * or DD DATA statement in the input stream:

```
//ddname  DD  *,DSID=xxxx,VOLUME=SER=yyyyyy
```

To merge associated data sets into the job input stream, the stream containing the DD statements for the associated data sets must be processed by the diskette reader program. JES2 and JES3 do not support the DSID parameter.

For more information on the 3540 diskette, see *3540 Programmer's Reference*.

Example

```
//ASSTDS  DD  DATA,DSID=3254,VOLUME=SER=778356
```

Identification through Catalog

A system or private catalog contains pointers to previously cataloged data sets. The system uses these pointers to locate data sets when a DD statement requests an old data set without UNIT or VOLUME parameters. For example:

```
//ddname  DD  DSNAME=dsname,DISP=OLD
```

Allocation and Unallocation of Catalog Volume

When the DSNAME parameter requests a cataloged data set, the system mounts the catalog volume, if it is not already mounted. From the catalog, the system obtains the pointer to the requested data set. Later, if the device on which the catalog is mounted is needed for another volume, the system demounts the catalog volume. The system assigns the catalog to the job step and performs disposition processing for the catalog volume when the job step ends.

In the following cases, the system does not mount the catalog volume during disposition processing of a job's data sets:

- The job abnormally terminates and data sets with an abnormal termination disposition of CATLG or UNCATLG were passed by a job step but not received by a later step.
- The system unallocates a step's data sets during warm start.

Using Private Catalogs

Private catalogs are defined on JOBCAT DD or STEPCAT DD statements. To define a private catalog, use access method services, as explained in *DFSMS/MVS Using Data Sets*. The system searches a private catalog before a system catalog when a JOBCAT or STEPCAT DD statement appears in the job or step and a DD statement does not specify unit and volume serial information for a data set. A JOBCAT catalog applies to each step of a job in which a STEPCAT catalog is not specified.

With SMS, do not use a JOBCAT DD statement in a job that references an SMS-managed data set and do not use a STEPCAT DD statement in a job step that references an SMS-managed data set. SMS only accesses SMS-managed data sets that are cataloged in a system catalog.

Note: In a JES3 system, a private catalog must be on a permanently resident volume.

To locate a data set, the system searches catalogs in the following order:

1. Private catalog(s) specified in the current step in a STEPCAT DD statement and statements concatenated to it.
2. If no private catalogs are specified for the job step, private catalogs specified in the current job in a JOBCAT DD statement and statements concatenated to it.
3. A CVOL indicated by the first qualifier, if any, of the data set name.
4. A private catalog indicated by the first one to four qualifiers, if any, of the data set name.
5. The system master catalog.

A private catalog can be either a VSAM user catalog or an integrated catalog facility catalog.

Examples

```
//CATDS   DD  DSNAME=DS1,DISP=OLD
//ANOTH   DD  DSNAME=A.B.C,DISP=OLD
//JOBCAT  DD  DSNAME=PRIVCAT1,DISP=SHR
//        DD  DSNAME=CONCAT2,DISP=SHR
//STEPCAT DD  DSNAME=PRIVCATS,DISP=SHR
```

Identification through Label

The system uses data set labels to:

- Identify volumes and the data sets they contain.
- Store data set attributes.

A label is either standard or nonstandard. Standard labels can be processed by the system; nonstandard labels must be processed by installation-written routines, which the installation adds to the system.

Data sets on tape volumes usually have labels; these labels can be standard or nonstandard. If labels are present, they precede each data set on the volume. Data sets on direct access volumes always have labels; these labels must be standard. Direct access labels are in the volume table of contents (VTOC) for the volume.

The label type subparameter tells the system the type of labels for the data set. The label type is coded:

```
//ddname  DD  LABEL=(,label)...
```

The label types are:

SL: IBM standard labels
SUL: both IBM standard and user labels
For data sets on direct access, only SL or SUL can be specified. For SL or SUL, or when the label type subparameter is omitted because the data set has IBM standard labels, the system ensures that the correct tape or direct access volume is mounted.

AL: ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels
AUL: ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels, and ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 user labels
For AL or AUL, the system ensures that the correct tape volume is mounted; the tape must have an ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 label.

NSL: nonstandard labels
For NSL, installation-provided nonstandard label processing routines must ensure that the correct tape volume is mounted.

NL: no labels
BLP: bypasses label processing
For NL or BLP, the operator must ensure that the correct tape volume is mounted. If you specify NL, the data set must not have any standard labels.

Use of BLP: BLP is not a label type, but a request that the system bypass label processing. Use this specification for a blank tape or for overwriting a seven-track tape at a parity or density different than its current parity or density.

LTM: bypasses a leading tape mark on unlabeled tape

Label Type for Cataloged or Passed Data Sets

For cataloged and passed data sets, the system does not keep label type information. Therefore, when referring to a cataloged or passed data set that has other than standard labels, code the LABEL type subparameter.

Nonspecific Volume Request

The label type subparameter can be specified for a nonspecific tape volume request, that is, a DD statement with no volume serial numbers. If the operator mounts a tape volume with a different label type, the system requests that the operator mount another volume. But, if the specified label type is NL or NSL for the nonspecific volume request and the operator mounts a volume with standard labels, the system uses the volume if **both** of the following are true:

1. The expiration date of the existing data set on the volume is passed.
2. The existing data set on the volume is not password protected.

If you specify SL on a nonspecific volume request, but the operator mounts a tape volume that contains other than IBM standard labels, the system asks the operator to identify the volume serial number and the volume's new owner before writing the

IBM standard labels. If the tape volume has ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels, the system asks the operator for permission to destroy the labels.

Specific Volume Request

If you specify SL on a specific volume request, that is, a DD statement that specifies volume serial numbers, but the volume does not contain IBM standard labels:

- If the mounted volume contains labels, the system rejects the volume and asks the operator to mount the specified tape volume.
- If the mounted volume is not labeled, the system asks the operator whether to reject the volume or write standard labels on it.

Examples

```
//DSF   DD  DSNAME=ALLAB,LABEL=(,AL),UNIT=3420,
//       VOLUME=SER=223344,DISP=(NEW,CATLG)

//DSJ   DD  DSNAME=CATDS,DISP=OLD,LABEL=(,SUL)
```

Identification by Location on Tape

When placing a data set on a tape volume that already contains one or more data sets, specify where the data set is to be placed, that is, whether the data set is to be second, third, fourth, etc., on the volume. Code the data set sequence number to position the tape:

```
//ddname  DD  LABEL=(data-set-sequence-number,label),...
//ddname  DD  LABEL=data-set-sequence-number,...
```

Data-Set-Sequence-Number with BLP

If you specify BLP for the label type, the system treats anything between tapemarks as a data set. Therefore, if the tape actually has labels, code the data-set-sequence-number subparameter to position the tape properly; the subparameter must reflect all labels and data sets that precede the desired data set. *DFSMS/MVS Using Magnetic Tapes* illustrates where tapemarks appear.

Examples

```
//DDEX1  DD  DSNAME=TAPEDS3,DISP=(NEW,KEEP),UNIT=3420,
//         LABEL=(3,SL),VOLUME=SER=666555

//DDEX2  DD  DSNAME=TAPEDS4,DISP=(NEW,KEEP),UNIT=3420,
//         LABEL=(8,BLP),VOLUME=SER=223344
```

Identification as TCAM Message Data Set

To identify a data set as containing telecommunications access method (TCAM) messages, code the following:

```
//ddname  DD  QNAME=procname
//ddname  DD  QNAME=procname.tcamname
```

The QNAME parameter refers to a TPROCESS macro instruction that defines a destination queue for the messages. The parameter can also name a TCAM job to process the messages.

Example

```
//EX1  DD  QNAME=MACRO1.TJOB
```

Identification as Data Set from or to Terminal (Non-APPC)

The TERM parameter has no function in an APPC scheduling environment. If you code TERM, the system will check it for syntax and ignore it.

In a job run in a TSO/E system, identify a data set as coming from or going to the terminal in the JOB statement USER parameter by coding:

```
//ddname  DD  TERM=TS
```

In a background or batch job, the system treats the TERM=TS parameter as a SYSOUT=* parameter if no other parameters are coded.

Example

```
//MYTSODS  DD  TERM=TS
```

Chapter 13. Data Set Resources - Description

Figure 13-1. Description Task for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Description					
of status	DISP				
of data attributes	DCB AMP DATACLAS KEYLEN DSNTYPE KEYOFF LRECL RECFM RECORG				
- by modeling	LIKE REFDD				
of migration and backup	MGMTCLAS				

Description of Status

The process of securing control of data sets for a job is called data set integrity processing. Data set integrity processing avoids conflict between two or more jobs that request use of the same data set. For example, two jobs, one named READ and another named MODIFY, both request data set FILE.

- READ wants only to read and copy certain records
- MODIFY deletes some records and changes other records

If both jobs use FILE concurrently, READ cannot be certain of the integrity of FILE because MODIFY is changing records in the data set. MODIFY should have exclusive control of the data set.

Indicate the type of control needed by coding the data set's status:

```
//ddname  DD  DISP=(NEW,...
//ddname  DD  DISP=(OLD,...
//ddname  DD  DISP=(MOD,...
//ddname  DD  DISP=(SHR,...
```

For exclusive use of a data set, code:

- NEW: the data set is being created in this job step.
- OLD: the data set existed before this job step.
- MOD: the system first assumes that the data set exists. For an existing sequential data set, MOD causes the read/write mechanism to be positioned

© Copyright IBM Corp. 1988, 1999

after the last record in the data set. The read/write mechanism is positioned after the last record each time the data set is opened for output.

If the system cannot find volume information for the data set on the DD statement, in the catalog, or passed with the data set from a previous step, the system assumes that the data set is being created in this job step. For a new data set, MOD causes the read/write mechanism to be positioned at the beginning of the data set.

Note: For a new generation of a generation data group (GDG) data set (where (+n) is greater than 0), VOLUME=REF or VOLUME=SER can be coded.

For shared use of a data set, code:

- SHR: the data set existed before this job step and can be read by other concurrent jobs.

Exclusive Control of a Data Set

When a job has exclusive control of a data set, no other job can use that data set until completion of the last **step** in the job that refers to the data set. A job should have exclusive control of a data set in order to modify, add, or delete records.

In some cases, you may not need exclusive control of the entire data set. You can request exclusive control of a block of records by coding the DCB, READ, WRITE, and RELEX macro instructions. See *DFSMS/MVS Using Data Sets* .

Shared Control of a Data Set

Several jobs can concurrently use a data set on a direct access device if they request shared control of the data set. None of the jobs should change the data set in any way.

If more than one step requests a shared data set, code SHR on every DD statement that requests the data set, if it is to be used by concurrently executing jobs.

Examples

```
//DD1   DD  DSNAME=PERMDS,DISP=OLD
//DD2   DD  DSNAME=&&TEMPDS,DISP=NEW
//DD3   DD  DSNAME=GENDS(+1),DISP=(NEW,CATLG)
```

Data Set Integrity Processing

The system performs data set integrity processing once for each job, for the following types of data sets:

- Permanent data sets
- Non-virtual I/O (VIO) temporary data sets
- Data sets with alias names, created with the access method services DEFINE command; see:

  DFSMS/MVS Access Method Services for ICF
  DFSMS/MVS Access Method Services for VSAM
- Members of generation data groups.

The system **does not** perform data set integrity processing for subsystem data sets.

Data Set Integrity Processing for Permanent Data Sets

To secure control for all **permanent data sets** for the job, the system enqueues each data set, marking the data set as requested by that job and noting the kind of control requested: shared or exclusive. The system assigns control of the data set until completion of the last step in the job that refers to the data set.

A statement requesting exclusive control overrides any number of statements requesting shared control. One of two methods can be used to request exclusive control:

- DISP=NEW, DISP=MOD or DISP=OLD on a JCL DD statement.
- DISP=NEW, DISP=MOD or DISP=OLD on a dynamic allocation request, including dynamic allocation requests that result from the use of certain utility control statements.

For example, utility control statements that delete/scratch a data set will result in exclusive use of that data set.

The job **receives** control of the data set if:

- Another job is not using the data set.
- Another job is using the data set but both the job requesting the data set and the job using the data set request shared control and no exclusive requests are pending.

The job does **not receive** control of a data set if:

- Another job is using the data set and that job has exclusive control.
- Another job is using the data set, with either exclusive or shared control, and this job requests exclusive control.
- Another job is using the data set, with shared control, and yet another, earlier job requests exclusive control.

If a job requests data sets that are not available, the system issues the message 'JOB jjj WAITING FOR DATA SETS' to the operator. The job waits until the required data sets become available, unless the operator cancels the job.

When the system has secured control of all permanent data sets, it allocates and unallocates resources for each step of the job. The job terminates after the system has unallocated all resources for the last step in the job.

Data Set Integrity Processing for Other Data Sets

Non-VIO temporary data sets, data sets with alias names, and members of generation data groups are reserved or enqueued for each step within the job. The job receives control of the data set for that step in the same way as for permanent data sets.

When each step terminates, the system releases control of any data sets that are not used in any subsequent step of the job, except non-VIO temporary data sets, data sets with alias names, or a member of a generation data group.

Summary of Data Set Integrity Processing

Figure 13-2. Data Set Integrity Processing

	Data set is currently in use:		Data set is not in use	Data set is previously requested for:	
	Shared control	**Exclusive control**		**Shared control**	**Exclusive control**
Permanent data set requested for:					
Shared control	Request granted	Request granted when data set released	Request granted	Request granted	Request granted when data set released
Exclusive control	Request granted when data set released	Request granted when data set released	Request granted	Request granted when data set released	Request granted when data set released
Non-VIO temporary data set requested for:					
Shared control	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Request granted	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx
Exclusive control	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx
GDG data set requested for:					
Shared control	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Request granted
Exclusive control	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx
Data set with alias name requested for:					
Shared control	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Request granted	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx
Exclusive control	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Request granted	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx	Fail or wait dependent on SDSN_WAIT specification in ALLOCxx

Description of Data Attributes

The system obtains information needed to read from and write to a data set from:

- The data control block (DCB).
- For a VSAM data set, from the access method control block (ACB).
- With SMS, from the data class of the data set.
- With SMS, from a model data set.

In Data Control Block (DCB)

The system obtains data control block information from the following sources, in override order:

- The DCB macro instruction, in assembler language programs, or file definition statements or language-defined defaults in programs in other languages.
- The DCB subparameters on the DD statement.

```
//ddname  DD  DCB=subparameter,...
//ddname  DD  DCB=(subparameter,subparameter,...),...
```

- The data set label.

Therefore, the system ignores a value in a DCB subparameter on the DD statement if the data control block already contains the value. The system ignores a value in the data set label if the data control block already contains the value from the program or a DD DCB subparameter.

Note: When concatenated data sets are involved, the DCB is completed based on the type of data set and how the processing program uses the data set. See *DFSMS/MVS Using Data Sets* for more information.

DCB Values from Cataloged Data Sets

The DD statement DCB parameter can ask the system to copy certain values from the data set label of a cataloged data set, by coding:

```
//ddname  DD  DCB=dsname,...
//ddname  DD  DCB=(dsname,subparameter,...)...
```

The system copies the DSORG, RECFM, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP values from the label. If any of these values are coded in subparameters following the dsname, the system uses the coded values.

DCB Values from Earlier DD Statements

The DD statement DCB parameter can ask the system to copy all subparameters from the DCB parameter in an earlier DD statement, by coding a backward reference to the earlier statement:

```
//ddname  DD  DCB=*.ddname
//ddname  DD  DCB=*.stepname.ddname
//ddname  DD  DCB=*.stepname.procstepname.ddname
```

Examples

```
//S1   EXEC  PGM=ANYA
//DD1  DD    DSNAME=ABC,DCB=(RECFM=FB,LRECL=80,BLKSIZE=960),
//     DISP=(NEW,CATLG,DELETE),UNIT=3380,VOLUME=223344,
//     SPACE=(CYL,(30,10))
//S2   EXEC  PGM=ANYB
//DD2  DD    DSNAME=COPIER1,DCB=ABC
//S3   EXEC  PGM=ANYC
//DD3  DD    DSNAME=COPIER2,DCB=*.S1.DD1
```

In Access Method Control Block (ACB)

The system obtains access method control block information for VSAM data sets from the following sources, in override order:

- The AMP subparameters on the DD statement.

  ```
  //ddname  DD  AMP=(subparameter),...
  //ddname  DD  AMP=('subparameter,subparameter,...'),...
  ```

- With SMS, the DD statement parameters KEYLEN, KEYOFF, LRECL, and RECORG.
- The ACB, EXLST, or GENCB macro instructions in assembler language programs.
- The catalog entry for the data set.

Therefore, the system ignores a value in a program macro instruction if the DD AMP parameter supplies the value. The system ignores a value in the data set catalog entry if the access method control block already contains the value from a DD AMP subparameter or a macro instruction in the program.

Note: The override order for ACB values is different from the override order for DCB values.

Examples

```
//DD4  DD  DSNAME=ANYVSAM1,AMP=('BUFND=4,BUFNI=4,STRNO=2'),
//     DISP=(NEW,CATLG,DELETE),UNIT=3380,VOLUME=556677,
//     SPACE=(TRK,(200,50))
```

In Data Class

With SMS, the system obtains information about the attributes of a data set from the data class for the data set.

In many cases, the attributes defined in the data class selected by an installation-written automatic class selection (ACS) routine are sufficient for the data sets you create with DD statements.

However, you can specify the name of a data class on the DATACLAS parameter for a new data set. (Note that an ACS routine can override the data class that you specify.)

The storage administrator at your installation defines the names of data classes and their data set attributes. To view a list of data class names and their attributes, use the Interactive Storage Management Facility (ISMF).

You can also override individual data set attributes. Any data set attributes you specify on the following parameters override the corresponding attributes in the data class for the data set.

RECORG (record organization) or RECFM (record format)
LRECL (record length)
KEYLEN (key length)
KEYOFF (key offset)
DSNTYPE (data set type, PDS or PDSE)
AVGREC (record request and space quantity)
SPACE (average record length, primary, secondary, and directory quantity)
RETPD (retention period) or EXPDT (expiration date)
VOLUME (volume-count)

Examples

```
//DD5  DD  DSNAME=DESIGNA.PGM,DISP=(NEW,KEEP)
//DD6  DD  DSNAME=DESIGNB.PGM,DATACLAS=PGM5,DISP=(NEW,KEEP)
//DD7  DD  DSNAME=DESIGNC.PGM,DATACLAS=PGM5,LRECL=1024,DISP=(NEW,KEEP)
```

From Model Data Set

With SMS, use the LIKE or REFDD parameter to copy data set attributes from a model data set:

- The LIKE parameter copies the attributes of an existing cataloged data set to the new data set that you are defining on a DD statement.
- The REFDD parameter copies the attributes of a data set that is defined in a previous DD statement to the new data set that you are defining on a DD statement.

Any data set attributes you specify on the DD statement that defines the new data set override the corresponding attributes copied from the model data set.

Examples

```
//DDEX  DD  DSNAME=DESIGN.EXMP,DISP=OLD
//DD8   DD  DSNAME=DESIGNE.PGM,LIKE=DESIGN.EXMP,DISP=(NEW,KEEP)
//DD9   DD  DSNAME=DESIGNF.PGM,LIKE=DESIGN.EXMP,LRECL=1024,
//          DISP=(NEW,KEEP)
//DD10  DD  DSNAME=DESIGNG.PGM,DATACLAS=DCLAS10,DISP=(NEW,KEEP)
//DD11  DD  DSNAME=DESIGNH.PGM,REFDD=*.DD10,LRECL=1024,
//          DISP=(NEW,KEEP)
```

Migration and Backup (with SMS)

For an SMS-managed data set (one with a storage class assigned), the system handles the migration and backup of the data set based on the attributes defined in the management class for the data set.

In many cases, the attributes defined in the management class selected by an installation-written automatic class selection (ACS) routine are sufficient for the data sets you create with DD statements.

However, you can specify the name of a management class on the MGMTCLAS parameter for a new SMS-managed data set. (Note that an ACS routine can override the management class that you specify.)

The storage administrator at your installation defines the names of management classes and their attributes. To view a list of management class names and their attributes, use the Interactive Storage Management Facility (ISMF).

Note that you cannot override any of the attributes defined in the management class for the data set.

Examples

```
//DD8  DD  DSNAME=DESIGND.PGM,DISP=(NEW,KEEP)
//DD9  DD  DSNAME=DESIGNE.PGM,MGMTCLAS=MCLASA,DISP=(NEW,KEEP)
```

Chapter 14. Data Set Resources - Protection

Figure 14-1. Protection Task for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Protection					
through RACF	PROTECT SECMODEL				
for ISO/ANSI/FIPS Version 3 tapes	ACCODE				
by passwords	PASSWORD and NOPWREAD on LABEL				
of access to BSAM and BDAM data sets	IN and OUT on LABEL				

Protection through RACF

To ask for RACF protection, code:

```
//ddname  DD  PROTECT=YES,...
```

or, with SMS:

```
//ddname  DD  SECMODEL=profile-name,...
```

Protection with the PROTECT Parameter

Through the PROTECT parameter, RACF can protect the following:

- A data set on a direct access volume
- A data set on a tape volume with labels, that is:

  LABEL=(,SL)
  LABEL=(,SUL)
  LABEL=(,AL)
  LABEL=(,AUL)
  LABEL=(,NSL) if the installation provides support

- A tape volume with or without labels, that is:

  LABEL=(,SL)
  LABEL=(,SUL)
  LABEL=(,AL)
  LABEL=(,AUL)
  LABEL=(,NSL)
  LABEL=(,NL)
  LABEL=(,BLP)
  LABEL=(,LTM)

© Copyright IBM Corp. 1988, 1999

For more information, see *OS/390 Security Server (RACF) Security Administrator's Guide*.

Examples

```
//TAPE2   DD  DSNAME=NEWDS1,PROTECT=YES,DISP=(NEW,KEEP),
//        VOLUME=(,,1,2,SER=(223344,556677)),
//        UNIT=(3400-5,2),LABEL=(,SUL)

//DISKDS  DD  DSNAME=NEWDS2,PROTECT=YES,DISP=(NEW,CATLG,KEEP),
//        VOLUME=SER=223344,UNIT=3380
```

Protection with the SECMODEL Parameter

With SMS, RACF can, through the SECMODEL parameter, protect a data set created under SMS.

You specify the name of a RACF data set profile on the SECMODEL parameter when you define a new data set. Use the SECMODEL parameter when you want to use a specific data set profile for a new data set rather than using your user/group default data set profile.

The data set profile contains information such as the name of the owner of the profile, a list of RACF users or groups authorized to access the data set, the access attempts that are logged, and other RACF-related information.

For more information, see *OS/390 Security Server (RACF) Security Administrator's Guide*, and *OS/390 Security Server (RACF) Command Language Reference*.

Example

```
//SMSDS  DD  DSNAME=NEWDS5.PGM,SECMODEL=(GROUP1.PROTA),DISP=(NEW,KEEP)
```

Protection for ISO/ANSI/FIPS Version 3 Tapes

To control access to an ISO/ANSI/FIPS Version 3 tape data set, code:

```
//ddname  DD  ACCODE=access-code,...
```

The system must contain an installation-written file-access exit routine. This routine verifies that the ACCODE parameter specifies the correct code for an existing data set and, therefore, can use a data set.

Examples

```
//DD1   DD  DSNAME=NEWDS,ACCODE=F,LABEL=(,AL),UNIT=3380,
//      VOLUME=SER=998877,DISP=(NEW,CATLG,KEEP)

//DD2   DD  DSNAME=OLDDS,ACCODE=J,LABEL=(,AL),UNIT=3380,
//      VOLUME=SER=665544,DISP=OLD
```

Protection by Passwords

Use the PASSWORD subparameter of the LABEL parameter to specify a password to be used for protecting a data set.

Note that SMS ignores the PASSWORD subparameter for SMS-managed data sets.

To protect a data set with a password, code:

```
//ddname DD LABEL=(data-set-sequence-number,label,PASSWORD)
//ddname DD LABEL=(data-set-sequence-number,,PASSWORD)
//ddname DD LABEL=(,,PASSWORD)
```

To use a password-protected data set, code:

```
//ddname DD LABEL=(data-set-sequence-number,label,PASSWORD)
//ddname DD LABEL=(data-set-sequence-number,,PASSWORD)
//ddname DD LABEL=(,,PASSWORD)
//ddname DD LABEL=(data-set-sequence-number,label,NOPWREAD)
```

These subparameters mean the following:

- PASSWORD: The data set cannot be read from, written to, or deleted by another job or step unless the operator supplies the system with the correct password.
- NOPWREAD: The data set cannot be written to or deleted by another job or step unless the operator supplies the system with the correct password. However, the data set can be read without the password.

To protect a data set with a password, specify PASSWORD when the data set is created. Password-protected data sets must have standard labels, either IBM standard or ISO/ANSI Version 1 or ISO/ANSI/FIPS Version 3 labels.

Examples

```
//EX1  DD  DSNAME=ABC,DISP=(NEW,CATLG,DELETE),
//     LABEL=(,SL,PASSWORD),UNIT=3400-5,VOLUME=223344

//EX2  DD  DSANME=DEF,DISP=OLD,LABEL=(,SL,NOPWREAD)
```

Protection of Access to BSAM or BDAM Data Sets

The LABEL parameter can modify the data set processing through the IN and OUT subparameters, as indicated in Figure 14-2, if the assembler OPEN macro instruction specifies the data set processing as:

- When using the basic sequential access method (BSAM): INOUT, OUTIN, OUTINX, or EXTEND
- When using the basic direct access method (BDAM): UPDAT

The LABEL subparameters are coded:

```
//ddname DD LABEL=(data-set-sequence-number,label,PASSWORD,IN)
//ddname DD LABEL=(,label,PASSWORD,OUT)
//ddname DD LABEL=(,,NOPWREAD,IN)
//ddname DD LABEL=(,,,OUT)
```

Figure 14-2. Processing with DD LABEL Subparameter IN or OUT

OPEN Macro Parameter	LABEL Subparameter	Program Processing of Data Set	Required Password
INOUT (BSAM) UPDAT (BDAM)	IN	Read records (If the program tries to write to the data set, the system gives control to the error analysis (SYNAD) routine.)	Read password, if data set protected with PASSWORD; write password, if data set protected with NOPWREAD
OUTIN (BSAM) UPDAT (BDAM)	OUT	Write records (If the program tries to read the data set, the system gives control to the error analysis (SYNAD) routine.)	Write password, if data set protected with PASSWORD or NOPWREAD
OUTINX (BSAM) EXTEND (BSAM)	OUT	Add records to end of data set (If the program tries to read the data set, the system gives control to the error analysis (SYNAD) routine.)	Write password, if data set protected with PASSWORD or NOPWREAD

Other Uses of the LABEL IN Subparameter

You can also use the IN subparameter to avoid operator intervention when reading a data set that has an unexpired expiration date.

Data Set Processing with LABEL OUT Subparameter

When the OPEN macro instruction specifies OUTINX or EXTEND and the DD LABEL contains an OUT subparameter, the system adds records to the end of the data set regardless of the DISP parameter of the DD statement.

Examples

```
//EX1  DD  DSNAME=D.E.F,DISP=OLD,LABEL=(,,NOPWREAD,IN)
//EX2  DD  DSNAME=EXIST,DISP=MOD,LABEL=(,,PASSWORD,OUT)
```

Chapter 15. Data Set Resources - Allocation

Allocation is the process the system uses to map requests for data sets to available devices and volumes. This chapter contains guidance information about the allocation of data set resources. Figure 15-1 shows the relationships between the allocation of resources associated with data sets, such as devices and volumes, and the appropriate JCL or JES statements and parameters.

Figure 15-1. Allocation Task for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Allocation					
of device	UNIT STORCLAS		CLASS on JOB (JES3 only)		SETUP and CLASS on //*MAIN
of tape or direct access volume	VOLUME STORCLAS				EXPDTCHK and RINGCHK on //*MAIN
of direct access space	SPACE AVGREC DATACLAS				
of virtual I/O	UNIT DSNAME= temporary data set				
with deferred volume mounting	DEFER on UNIT				
with volume premounting				/*SETUP	
dynamic			DYNAMNBR on EXEC		

This chapter includes the following topics related to the allocation of data set resources.

- "Allocation of Device" on page 15-2
- "Allocation of Volume" on page 15-16
- "Interactions Between Device and Volume Allocation" on page 15-24
- "Stacking Data Sets" on page 15-39
- "Allocation of Direct Access Space" on page 15-44
- "Allocation of Virtual I/O" on page 15-49
- "Allocation with Volume Premounting in a JES2 System" on page 15-52
- "Dynamic Allocation" on page 15-53

Some of these topics include sections that describe the topic from the perspective of whether the resource is SMS-managed or non-SMS-managed.

© Copyright IBM Corp. 1988, 1999

In this chapter, **SMS-managed** and **system-managed** are used interchangeably to describe resources that the storage management subsystem (SMS) manages, and **with SMS** indicates information that applies when SMS is installed and active.

Data sets on system-managed tape library volumes exhibit both system-managed and non-system-managed characteristics. When necessary, **datasets on a system-managed tape volume** are distinguished from **system-managed DASD data sets**. Otherwise, the term **system-managed data sets** refers to both datasets on a system-managed tape volume and system-managed DASD data sets.

Allocation of Device

The device that a data set resides on is determined as follows:

- **For SMS-managed data sets**, by the storage class for the new data set, specified on the STORCLAS parameter of the DD statement or selected by the installation-written automatic class selection (ACS) routine for the new data set.
- **For non-SMS-managed data sets**, by the UNIT parameter, specified on the DD statement for the new data set, or, with SMS, by the SMS default unit, when the UNIT parameter is not specified.

Device Allocation for SMS-Managed Data Sets

For an SMS-managed data set, SMS obtains information about the device to be used for the data set based on the storage class assigned for the data set.

In many cases, the device used by the storage class that an ACS routine selects is sufficient for the data sets you create with DD statements.

You can, however, specify the name of a storage class on the STORCLAS parameter for a new SMS-managed data set. (Note that an ACS routine can override the storage class that you specify.)

The storage administrator at your installation defines the names of storage classes and their attributes. To view a list of storage class names and their attributes, use Interactive Storage Management Facility (ISMF).

To let an ACS routine select a storage class for a new data set, omit the STORCLAS parameter; for example:

```
//DD5  DD  DSNAME=DESIGNA.PGM,DISP=(NEW,KEEP)
```

To specify a specific storage class for a new data set, code the STORCLAS parameter; for example:

```
//DD6  DD  DSNAME=DESIGNB.PGM,STORCLAS=STOR55,DISP=(NEW,KEEP)
```

The system catalogs new permanent system-managed DASD data sets at allocation. The system catalogs datasets on a system-managed tape volume during unallocation processing, according to DISP parameters on DD statements.

To retrieve an existing data set, you do not need to code the STORCLAS parameter; for example:

```
//DD7  DD  DSNAME=DESIGNB.PGM,DISP=MOD
```

If you specify the UNIT parameter for an SMS-managed data set, the system generally ignores the parameter. There are, however, several cases when the system uses the information specified on the UNIT parameter:

- For system-managed DASD data sets, the system honors the unit-count subparameter but ignores all other subparameters on the UNIT parameter.
- For datasets on a system-managed tape volume, the system ignores the device type, device number, and group name subparameters on the UNIT parameter but honors all other subparameters on the UNIT parameter.

Device Allocation for Non-SMS-Managed Data Sets

On the DD statement for a non-SMS-managed data set, code a UNIT parameter to indicate the device on which the data set resides or is to be written.

With SMS, you do not need to code the UNIT parameter if the storage administrator at your installation has defined a system default unit to be used for new data sets. Check with your storage administrator.

The UNIT parameter can specify:

- A particular device:

```
//ddname DD  UNIT=device-number,...
```

- A type of device, such as a 3350 direct access device or a 1403 printer:

```
//ddname DD  UNIT=device-type,...
```

- A group of devices, such as DISK, to indicate all direct access devices in the system:

```
//ddname DD  UNIT=group-name,...
```

The status of a device affects whether the system can allocate it or not. See Figure 15-2.

Figure 15-2. Effect of Device Status on Allocation

Status	Device Type				
	Direct Access	**Tape**	**Printer Punch**	**Graphic**	**Teleprocessing**
Online	Eligible for allocation				
Offline	Eligible for allocation when the operator brings the device online				Eligible for allocation when at least one path to the device is online
Pending Unload	Eligible for allocation when the volume is specifically requested		Not applicable		
Pending Offline	Eligible for allocation when the operator selects the device in response to message IEF238D or when the operator brings the device online.		Eligible for allocation when the operator selects the device in response to message IEF238D or when the operator brings the device online.		Not applicable

Specifying Device Number

The **device number** is a 3-digit or 4-digit hexadecimal number assigned to the device when it is installed. In JCL statements, always precede a 4-digit number with a slash (/). A 3-digit number can be specified with or without a slash.

A 3-digit device number can be specified in two formats, where `h` is a hexadecimal digit:

- 3-digit format: `hhh` or `/hhh`
- 4-digit format: `/0hhh`

Note that the slash before a 4-digit device number distinguishes it from a device type, which is also 4 digits, but cannot contain a slash or be preceded by a slash.

For example, UNIT=/3490 is the device number for a specific device.

Do not specify a device by its number unless absolutely necessary. When you specify a device number, the system can assign only that specific device. Specifying a device number will delay a job if another job is using the device.

Specifying Device Type

Requesting a **device type** allows the system to assign any available device of that type. For example, UNIT=3350 indicates that you want the system to assign any available 3350 Direct Access Storage device. For more information on specifying device types, see *OS/390 HCD Planning*.

Specifying Group Name

During system initialization, the installation can define **group names** for a group of devices. The devices in a group may or may not all be the same type. Requesting a group name allows the system to assign any available device in the group. For example, if the group named DISK includes 3350 and 3380 Direct Access Storage devices, the system assigns an available 3350 or 3380 device when UNIT=DISK is coded. If the group named 3350A includes three particular 3350 devices, the system assigns one of these 3350 devices when UNIT=3350A is coded.

Groups with Several Types of Devices

If the group contains more than one type of device and the DD statement requests more than one device, the system allocates devices of the same type from the group. For example, if the group named TAPE includes both 3400-5 and 3400-6 devices and the DD statement specifies UNIT=(TAPE,2), the system assigns either two 3400-5s or two 3400-6s. If the system does not have enough devices of one type to satisfy the request, the system terminates the job.

If a group contains more than one type of device, do not code the group name when requesting an existing data set or a specific volume. The system may assign one type of device while the data set resides on another type. For example, if SYSSQ contains all tape and direct access devices, do not code UNIT=SYSSQ for an existing data set on tape; the system might assign a direct access device.

Groups with Devices with Special Features

This rule also applies if the data set resides on a 3348 Model 70F Data Module and the group name includes 3340 drives with and without the Fixed Head Feature. The 3348 Model 70F must be assigned to a 3340 with the feature. For more information on the Fixed Head Feature, see the *IBM 3340 Disk/Storage - Fixed Head Feature User's Guide*.

If a nonspecific volume request requires more than one tape device from a group that contains both single and dual density tape drives, the system assigns the devices so that the single density drive is the first one used. The default density is the density of the single density drive. The operator may be requested to mount the volumes in a different order than assigned by the system.

Concurrent Allocation of Devices

Only direct access devices can be allocated to different jobs executing concurrently. Teleprocessing equipment cannot be allocated more than once in the same job step. If a printer, punch, teleprocessing equipment, or graphics device is designated as a console, it cannot be allocated to a job.

Allocating a Teleprocessing Device With a Group Name

If you request that the system allocate one or more lines of a line group by using a group name, the system attempts to allocate the lines within the line group, starting with the lowest teleprocessing (TP) line address and continuing in ascending order. If the first eligible line in the line group is already allocated, the system fails the request to allocate from that line group.

Note: A group name is called an esoteric name in Hardware Configuration Definition (HCD) terminology.

Definition of UNIT Parameters in System Initialization

The installation describes each device to the system during system initialization. During this process, the installation defines the device types and group names to be coded in the DD UNIT parameter.

The installation should maintain a list of the device types and group names. For more information, see *OS/390 HCD Planning*.

Specifying Device for Output Data Set (Non-SMS-Managed Data Sets)

To print or punch a data set without using the job entry subsystem output service, specify the printer or punch in the UNIT parameter on the DD statement for the data set. The system allocates the device, if available, exclusively to the job; jobs cannot share output devices. Data management routines write the output from the program to the specified device.

Sending output through the job entry subsystem to a sysout data set is usually more efficient. JES uses the printers and punches for many jobs without intermixing output.

Allocation with Deferred Volume Mounting

A step can include a data set that the program might not use. To ask the system not to mount the volume for the data set until the data set is opened, code:

```
//ddname  DD  UNIT=(xxxx,,DEFER),...
```

Deferred mounting can save the operator time.

Example

```
//MYDS  DD  DSNAME=DATA5,UNIT=(TAPE,,DEFER)
```

Note: You can also use deferred mounting for SMS-managed data sets.

Requesting More than One Unit for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

For faster processing, request several units for a multivolume data set or for a data set that may require additional volumes. When each volume is on its own device, step execution is not halted while the operator demounts and mounts volumes.

Always request several units when the data set resides on more than one permanently resident or reserved volumes or may be extended to a new volume during step execution. Permanently resident and reserved volumes cannot be demounted in order to mount a new volume.

Request multiple units by:

- Coding the unit count subparameter:

  ```
  //ddname  DD  UNIT=(device,unit-count),...
  ```

- Requesting parallel mounting when the VOLUME parameter requests more than one volume in the volume count parameter or in more than one serial number:

  ```
  //ddname  DD  UNIT=(device,P),VOLUME=(,,,volume-count)
  //ddname  DD  UNIT=(device,P),
  //          VOLUME=SER=(serial-number,serial-number,...)
  ```

Number of Devices Allocated for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

The system assigns volumes and devices for a job step by calculating the following:

- The maximum number of volumes per DD statement
- The maximum number of devices per DD statement
- The number of devices for the step

Volumes Required per DD Statement

See “Volumes Required per DD Statement for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume” on page 15-23.

Devices Required per DD Statement

The maximum number of tape devices or direct access devices required to satisfy any DD statement is the unit count in the UNIT parameter except when volume affinity is present. If volume affinity is present, the number of devices might be more than the unit count in the UNIT parameter. For more information, see “Devices Assigned per Step” on page 15-7.

However, if the UNIT parameter also specifies P, for parallel mount, the system uses the **greatest** of the following numbers to determine how many devices and volumes to allocate:

- Unit-count in the UNIT parameter
- Volume-count specified in the VOLUME parameter
- Number of serial numbers implicitly or explicitly specified
- With SMS, volume-count in the data class

The number of devices is affected by the DD statement parameters as follows:

DD Statement Specifies	System Action
UNIT=AFF	The system obtains the device requirements from the referenced DD statement. All of the devices used for the referenced DD statement are shared with the referring statement's data set.
Generation data group (GDG)	The system determines the number of devices needed by totaling the devices needed for each generation data set. Each generation data set is handled as a single request.
VSAM data set	The system determines the number of devices needed based on the device/volume configuration of the data set. If the data set is on more than one type of device, the system determines the total number of devices required and allocates them. The system may override the unit count or parallel mounting, if specified.
Unit name that includes different device types	The system allocates devices of the same type.

Devices Assigned per Step

The number of devices assigned for a job step is not necessarily the sum of the device requirements for each DD statement.

The following tend to reduce the total devices assigned for a step:

- A volume can be allocated to only one device. Therefore, when more than one DD statement asks for the same volume, the system allocates the same volume on the same device.
- Requests for direct access space on public and/or storage volumes can be allocated to the same volume. Therefore, when more than one DD statement requests such space, the system can allocate the same volume on the same device.
- Requests for the same public tape volume are allocated to that volume. Therefore, if a DD statement requests a public tape and specifies VOLUME=REF, the system can allocate the same volume on the same device.

The following tend to increase the total devices assigned for a step:

- A permanently resident or reserved volume cannot be demounted. Therefore, the system assigns a permanently resident or reserved volume to its own device, on which it is mounted. The volume is assigned to its own device even

if the DD statements specify that the device was to be shared with other volumes.

- A direct access volume is requested by more than one DD statement in a step; the volume is shared by the data sets. The system assigns that volume to a device and does not assign any other volumes to that device, even if the DD statements specify that the device was to be used for other volumes.
- The system allocates additional devices for a VSAM data set, if the data set resides on more than one type of device.
- The system allocates a direct access device for a private catalog, if it is associated with and/or used to retrieve volume information about a requested data set.
- For a generation data group (GDG), the system may have to assign additional devices to satisfy the device type needs for each generation data set in the GDG.
- When DD statements request conflicting device assignments for a tape volume, the system assigns the volume involved in the conflict its own device. For example:

```
//DD1  DD  UNIT=2400,VOLUME=SER=(V1,V2)
//DD2  DD  UNIT=2400,VOLUME=SER=(V2,V3)
```

  Volume serial V2 has conflicting device assignments. Therefore, the system assigns the three volumes to three devices. If the DD2 had requested unit affinity, UNIT=AFF=DD1, the system would have assigned only one device to all three volumes.

Examples for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

Example 1

```
//TEST   JOB  5675,'DEPT. 25'
//STEP1  EXEC PGM=A1
//D1     DD   DSNAME=A01DD1,DISP=(,PASS),UNIT=3330,
//         SPACE=(TRK,1),VOLUME=SER=333001
//STEP2  EXEC PGM=A2
//D2     DD   DSNAME=LIB1,DISP=OLD,UNIT=3340,
//         VOLUME=(PRIVATE,SER=123456)
//D3     DD   DSNAME=ABC,DISP=(OLD,KEEP),UNIT=AFF=D2,
//         VOLUME=SER=777777
//D4     DD   DSNAME=TAPE,DISP=OLD,UNIT=(3420-5,P,DEFER),
//         VOLUME=SER=(342001,342002,342003,342004,342005)
//D5     DD   DSNAME=DISK,DISP=(SHR,KEEP),UNIT=(,P),
//         VOLUME=SER=(333005,333008,333010)
//D6     DD   UNIT=3340,VOLUME=REF=*.D2,SPACE=(TRK,(5,2))
//D7     DD   UNIT=3340,VOLUME=REF=DISK,SPACE=(TRK,(10,5))
```

- D1 defines a new data set named A01DD1. It is to be on volume 333001, which is mounted on a 3330 Disk Storage.
- D2 defines an old data set named LIB1, which resides on a private volume, 123456. The volume is mounted on a 3340 Direct Access Storage.
- D3 defines an old data set named ABC. This data set is to be kept after this step terminates. ABC is on volume 777777. This volume is to be mounted on the same 3340 device used for D2.

- D4 defines an old data set named TAPE. The data set is on the five volumes identified in the VOLUME parameter. The DEFER subparameter indicates that the five volumes are to be mounted only after the data set is opened. The P subparameter requests parallel mounting; that is, all five volumes are to be mounted at the same time on five different 3420-5 Magnetic Tape Units.
- D5 defines an old data named DISK. This data set can be shared by another job; the program only reads it. The data set is to be kept after this step. The system determines the number of devices to be allocated from the number of volume serials requested: in this case, three.
- D6 is a temporary data set, which is indicated by omission of a DSNAME parameter. The system, therefore, assumes a disposition of NEW,DELETE. The system is to place the data set on the volume used for D2 in STEP2, that is, volume 123456.
- D7 is also a temporary data set. The backward reference for volume information is to the dsname DISK, which was defined in D5 in STEP2. The system is to place this data set on the three volumes 333005, 333008, and 333010.

Example 2

```
//STEPA  EXEC PGM=TESTA
//A1     DD   UNIT=3400-5,VOLUME=SER=111111
//A2     DD   UNIT=AFF=A1,VOLUME=SER=222222
```

The system assigns one unit for both volumes. Volume 111111 is mounted first; 222222 is mounted when A2 is opened. This processing is the same for both tape and direct access.

Example 3

```
//STEPB  EXEC PGM=TESTB
//B1     DD   UNIT=(3330,2),VOLUME=SER=(A,B)
//B2     DD   UNIT=AFF=B1,VOLUME=SER=(C,D)
```

The system allocates two units to B1; volumes A and B are mounted. B2 gets allocated to the same two units; volumes C and D are mounted when the data set for B2 is opened.

Example 4

```
//STEPC  EXEC PGM=TESTC
//C1     DD   UNIT=(3330,2),VOLUME=SER=(A,B)
//C2     DD   UNIT=AFF=C1,VOLUME=SER=(C,D)
//C3     DD   UNIT=3330,VOLUME=SER=B
```

STEPC shows a direct access example of volume affinity for volume B. The system allocates volumes A and C to share one unit and volumes B and D to two other units.

Example 5

```
//STEPD  EXEC PGM=TESTD
//D1     DD   UNIT=(3330,2),VOLUME=SER=(E,F)
//D2     DD   UNIT=AFF=D1,VOLUME=SER=(G,H)
```

STEPD is a direct access example. If volume E is currently mounted and is permanently resident or reserved, the system allocates a separate unit for volume E because it cannot be dismounted. The system allocates one unit for volume G and a second unit to be shared by volumes F and H. Therefore, three volumes are used, instead of two, because of the permanently resident or reserved attributes.

Example 6

```
//STEPE  EXEC PGM=TESTE
//E1     DD   UNIT=3400-5,VOLUME=SER=(111111,222222)
//E2     DD   UNIT=AFF=E1,VOLUME=SER=(222222)
```

STEPE is a tape example. The system allocates two units: one for volume 111111 and the second for volume 222222. Note that only one data set can be open on a tape volume at a time; to prevent an error when the data set for E2 is opened, the data set for E1 must be closed before E2 is opened.

Example 7

```
//STEPF  EXEC PGM=TESTF
//F1     DD   UNIT=3330,VOLUME=SER=(ABCDEF,GHIJKL)
//F2     DD   UNIT=AFF=F1,VOLUME=SER=(ABCDEF)
```

STEPF is a direct access example. The system ignores the volume affinity between F1 and F2. Volume ABCDEF of both DD statements uses one unit while the other volume, GHIJKL, uses a different unit.

Example 8

```
//STEPG  EXEC PGM=TESTG
//G1     DD   UNIT=3400-5,VOLUME=SER=111111
//G2     DD   UNIT=AFF=G1,VOLUME=SER=111111
//G3     DD   UNIT=AFF=G1,VOLUME=SER=222222
```

In STEPG, G2 and G3 request unit affinity to G1. The system allocates one unit to be used for volume 111111 and volume 222222.

Example 9

```
//STEPH EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=INPUT.DATASET,DISP=SHR
//SYSUT2   DD DSN=OUTPUT.DATASET,DISP=(NEW,KEEP),LABEL=(1,SL),
//     STORCLAS=LIBRARY,DATACLAS=PITTBRGH
```

STEPH copies an input data set to a new output data set on a system-managed tape volume to be shipped offsite to Pittsburgh. The output data set is directed to a system-managed tape library because of the storage class "LIBRARY".

Data class "PITTBRGH" defines the media type and recording format requirements of the Pittsburgh data center. If either the media type or the recording-format requirements of that center changes, the storage administrator modifies the "PITTBRGH" data class definition but does not have to modify JCL.

Example 10

```
//STEPI EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=INPUT.PAYROLL,DISP=SHR
//SYSUT2   DD DSN=OUTPUT.PAYROLL,DISP=(NEW,KEEP),LABEL=(1,SL),
//     DATACLAS=PAYROLL
```

STEPI copies an input payroll data set to a data set on a system-managed tape volume. The installation's ACS routines must assign a storage class to DD SYSUT2 that directs the allocation to a system-managed tape library. The data class "PAYROLL" defines the media and record format required for payroll data. If either the media type or recording format requirements for payroll data changes, the storage administrator modifies the "PAYROLL" data class definition but does not have to modify JCL.

Example 11

```
//STEPJ EXEC PGM=IEBCOPY
//ICOPY001 DD DISP=SHR,DSN=DASD.DS1
//OCOPY001 DD UNIT=(3490,,DEFER),DISP=(,KEEP),
//         DSN=USERID.TEST1.ATL,VOL=(,RETAIN)
//ICOPY002 DD DISP=SHR,DSN=DASD.DS2
//OCOPY002 DD UNIT=AFF=OCOPY001,DISP=(,KEEP),LABEL=2,
//         DSN=USERID.TEST2.ATL,VOL=(,RETAIN,REF=*.OCOPY001)
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
          COPY  OUTDD=OCOPY001,INDD=ICOPY001
          COPY  OUTDD=OCOPY002,INDD=ICOPY002
/*
```

This example shows data set stacking using VOL=REF. STEPJ stacks copies of DASD data sets represented by ICOPY001 and ICOPY002 onto an output system-managed tape volume defined by statements OCOPY001 and OCOPY002. Because these data sets will be opened serially, only one system-managed tape library device needs to be allocated.

The installation's ACS routines must assign a storage class that directs the allocation of DD OCOPY001 to a system-managed tape library (OCOPY002 assumes the library status of OCOPY001 by its volume reference). Because OCOPY002 specifies unit affinity to DD OCOPY001, the system allocates only one system-managed tape library device for these two DD statements.

For more information about data set stacking, see “Stacking Data Sets” on page 15-39.

Example 12

```
//STEPK EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1  DD DSN=INPUT.18TRACK.LIBRARY.DATASET,DISP=SHR,LABEL=(,,,IN)
//SYSUT2  DD DSN=OUTPUT.DATASET,DISP=(NEW,PASS),LABEL=(1,SL),
//         STORCLAS=LIBRARY,DATACLAS=PITTBRGH
```

STEPK copies existing data set INPUT.18TRACK.LIBRARY.DATASET to new data set OUTPUT.DATASET. Because the existing data set was recorded on an

18-track format device, and will not be extended during the allocation of DD SYSUT1, the system can use any device that can read an 18-track formatted volume for the allocation.

If the IBM 3495 Tape Library Dataserver contains both 3480X devices (18-track read/write) and 3490 devices (18-track and 36-track read, 36-track write), using LABEL=(,,,IN) to allocate SYSUT1 means that either device can be allocated.

Device Allocation in a JES3 System

In a JES3 system, the devices and volumes for each data set are allocated by JES3 or the system.

Device Management

Allocation of a device depends on whether it is managed by MVS, by JES3, or jointly by JES3 and MVS. Device management is shown in the following chart.

Management	Devices
By MVS	Any devices not defined to JES3 during JES3 initialization
Jointly by JES3 and MVS	Direct access with permanently resident or reserved volumes: By JES3 for specific volume requests or for private volumes By MVS for nonspecific volume requests or for public or storage volumes
By JES3	Direct access with removable volumes: Tape devices Printers Punches Graphic devices

During JES3 initialization, the installation defines how each device is to be managed. See *OS/390 JES3 Initialization and Tuning Guide*.

Device Allocation

JES3 allocates JES3-managed devices and jointly-managed devices; JES3 performs all allocation before the job is initiated for execution. MVS allocates MVS-managed devices and jointly-managed devices; MVS performs all allocation when a step is being initiated for execution.

For a JES3-managed device, you can change the way JES3 handles allocation by coding:

```
//*MAIN  SETUP=JOB
//*MAIN  SETUP=HWS
//*MAIN  SETUP=THWS
//*MAIN  SETUP=DHWS
//*MAIN  SETUP=(stepname.ddname,...)
//*MAIN  SETUP=(stepname.procstepname.ddname,...)
//*MAIN  SETUP=/(stepname.ddname,...)
//*MAIN  SETUP=/(stepname.procstepname.ddname,...)
```

Effect of Job Class on Allocation

The CLASS parameter has no effect in an APPC scheduling environment. If you code CLASS in that environment, the system will check the parameter for syntax and ignore it.

The job class affects which devices can be allocated to the job. During JES3 initialization, the installation identifies the execution resources, including devices, that can be assigned to each job class.

The job class is specified by coding one of the following; if neither is coded, the system assigns the job to the installation-defined standard default class.

```
//jobname JOB  acct,progname,CLASS=jobclass
//*MAIN CLASS=class-name
```

Catalog Use

For allocation, JES3 accesses the catalog at job setup time, whereas MVS accesses the catalog at step initiation time. After job setup and before step initiation, the catalog can be changed by, for example, an IBM utility, user utility, or system routine. Because JES3 and MVS access the catalog at different times, catalog changes can cause unpredictable results. Therefore, the installation should not change the catalog while jobs are being scheduled.

Types of JES3 Setup

JES3 allocates devices in three different ways: job setup, high watermark setup, and explicit setup. The type of setup to be used is specified during JES3 initialization, but can be changed for a job by parameters on the //*MAIN statement.

Job setup

For job setup, JES3 allocates all the JES3-managed and jointly-managed devices required in the job before the job is initiated. JES3 mounts the initial volumes necessary to run all steps before the job executes. To request job setup, code:

```
//*MAIN  SETUP=JOB
```

When volumes are no longer needed, they are demounted, if removable, and the devices unallocated, that is, made available for use by another job. If you specify the FREE=CLOSE DD parameter, JES3 unallocates the device when the data set is closed.

If you are using the dequeue at demount facility (early volume release) for multivolume data sets, JES3 unallocates volumes when they are demounted. For information on the dequeue at demount facility, see the TYPE=J OPEN macro option in *DFSMS/MVS DFSMSdfp Advanced Services*.

Figure 15-3. JES3 Job Setup (SETUP=JOB)

Devices and Volumes to be Allocated	**Tape**						**Direct Access**				
Volumes on Devices Set Up Before Execution	1	2	3	4	5	6	8	9	10	11	12
Job Steps **STEP 1** tape volume=1,2 direct access volume=8,9	U	U	A	A	A	A	U	U	A	A	A
STEP 2 tape volume=2,3,4 direct access volume=8	A	U	U	U	A	A	U	A	A	A	A
STEP 3 tape volume=4 direct access volume=9,10,11	A	N	N	U	A	A	A	U	U	U	A
STEP 4 tape volume=1,5,6 direct access volume=8,11,12	U	N	N	N	U	U	U	N	N	U	U
Total Devices Used by the Job for Setup	6 Tape						5 Direct Access				

Legend

U The device is allocated and in **use**

A The device is **allocated** but not in use

N The device is **no** longer **needed** and can be unallocated.

High Watermark Setup

For high watermark setup, JES3 reserves for a job the maximum number of devices of each type needed for any one job step. JES3 premounts only some volumes before the job executes. When you must use fewer devices for a job, high watermark setup is better than job setup. To request high watermark setup, code:

- High watermark setup for tapes, direct access, graphics, printers, and punches:

  ```
  //*MAIN SETUP=HWS
  ```

- High watermark setup for tapes only, with job setup for direct access:

  ```
  //*MAIN SETUP=THWS
  ```

- High watermark setup for direct access, with job setup for tapes:

  ```
  //*MAIN SETUP=DHWS
  ```

When the last step that uses a device no longer needs it, JES3 unallocates it.

In Figure 15-4, volumes mounted after STEP1 are indicated by placing the volume number in the box for the step in which it is allocated. For example, Volume 3 is mounted at STEP2.

Figure 15-4. JES3 High Watermark Setup (SETUP=HWS)

Devices and Volumes to be Allocated	**Tape**			**Direct Access**		
Volumes on Devices Set Up Before Execution	1	2	4	8	9	11
Job Steps **STEP 1** tape volume=1,2 direct access volume=8,9 Volume 1 is mounted at STEP1 and then demounted until needed in STEP4. Volume 8 is mounted for STEP1 and STEP2 and then demounted until needed in STEP4.	U	U	A	U	U	A
STEP 2 tape volume=2,3,4 direct access volume=8 Volume 3 is mounted at STEP 2.	U 3	U	U	U	A	A
STEP 3 tape volume=4 direct access volume=9,10,11 Volume 10 is mounted at STEP 3.	A	A	U	U 10	U	U
STEP 4 tape volume=1,5,6 direct access volume=8,11,12 Volumes 1, 5, 6, 12, and 8 are mounted at STEP 4. Volumes 1 and 8 are mounted on any available device.	U 1	U 5	U 6	U 12	U 8	U
Total Devices Used by the Job for Setup	3 Tape			3 Direct Access		

Legend

U The device is allocated and in **use**

A The device is **allocated** but not in use

N The device is **no** longer **needed** and can be unallocated.

Explicit setup

Explicit setup is directed by the user. Explicit setup requires the same number of devices as job setup. JES3 premounts volumes according to the instructions coded in:

```
//*MAIN  SETUP=(stepname.ddname,...)
//*MAIN  SETUP=(stepname.procstepname.ddname,...)
```

To request that JES3 not explicitly set up certain volumes, code:

```
//*MAIN  SETUP=/(stepname.ddname,...)
//*MAIN  SETUP=/(stepname.procstepname.ddname,...)
```

The advantage of explicit setup over high watermark setup is that you can force volumes to stay mounted on devices until they are no longer needed. The disadvantage is that JES3 does not unallocate devices early: JES3 allocates a certain number of devices before job execution and does not unallocate any until the job completes execution. In contrast, with job setup and high watermark setup, JES3 can unallocate devices at the end of any step, if the devices are no longer needed.

In the explicit setup shown in Figure 15-5 on page 15-16, four devices are allocated for both tape and disk instead of the three allocated using high watermark

setup. The volumes to be explicitly mounted, for example, volumes 1 and 8, are not unallocated and then remounted for the last step.

Figure 15-5. JES3 Explicit Setup (SETUP=ddname)

Devices and Volumes to be Allocated	**Tape**				**Direct Access**			
Volumes on Devices Set Up Before Execution	1	2	3	4	8	9	10	11
Job Steps **STEP 1** tape volume=1,2 direct access volume=8,9	U	U	A	A	U	U	A	A
STEP 2 tape volume=2,3,4 direct access volume=8	A	U	U	U	U	A	A	A
STEP 3 tape volume=4 direct access volume=9,10,11	A	A	A	U	A	U	U	U
STEP 4 tape volume=1,5,6 direct access volume=8,11,12 Volumes 5, 6, and 12 are mounted in STEP 4.	U	A	U 5	U 6	U	A	U 12	U
Total Devices Used by the Job for Setup	4 Tape				4 Direct Access			

Legend

U The device is allocated and in **use**

A The device is **allocated** but not in use

N The device is **no** longer **needed** and can be unallocated.

Altering JES3 Device Allocation

To keep JES3 from allocating devices before the first step and holding them until a later step needs them, break a multiple-step job into several smaller jobs in a dependent job net.

Allocation of Volume

The volume that a new data set resides on is determined as follows:

- **For system-managed DASD data sets**, either by the:
  - Storage class for the new data set, specified on the STORCLAS parameter of the DD statement or selected by an installation-written automatic class selection (ACS) routine.
  - VOLUME parameter, specified on the DD statement for the new data set if the storage class is GUARANTEED_SPACE=YES.
- **For datasets on a system-managed tape volume**, either by the:
  - Storage class for the new data set, specified on the STORCLAS parameter of the DD statement or selected by an installation-written automatic class selection (ACS) routine.
  - VOLUME parameter, specified on the DD statement for the new data set.
- **For non-system-managed data sets**, by the VOLUME parameter, specified on the DD statement for the new data set.

Volume Allocation for SMS-Managed Data Sets

For an SMS-managed data set, the system uses the storage class to select a volume or volumes for the data set.

In many cases, you can allow an ACS routine to assign a storage class to the data set and allow SMS to select the volume(s) based on the storage class.

You can, however, specify the name of a storage class on the STORCLAS parameter for a new SMS-managed data set. (Note that an ACS routine can override the storage class that you specify.)

The storage administrator at your installation defines the names of storage classes and their attributes. To view a list of storage class names and their attributes, use Interactive Storage Management Facility (ISMF).

To let an ACS routine select a storage class for a new data set, omit the STORCLAS parameter; for example:

```
//DD10  DD  DSNAME=DESIGNF.PGM,DATACLAS=DCLAS10,DISP=(NEW,KEEP)
```

To specify a specific storage class for a new data set, code the STORCLAS parameter; for example:

```
//DD11  DD  DSNAME=DESIGNG.PGM,DATACLAS=DCLAS12,STORCLAS=STOR55,
//        DISP=(NEW,KEEP)
```

The system catalogs new permanent system-managed DASD data sets at allocation. The system catalogs datasets on a system-managed tape volume during unallocation processing, according to DISP parameters on DD statements.

To retrieve an existing data set, you do not need to code the STORCLAS parameter; for example:

```
//DD12  DD  DSNAME=DESIGNG.PGM,DISP=MOD
```

References to SMS-Managed Data Sets

If you specify VOLUME=REF and refer to an SMS-managed data set, SMS manages the new data set using the same storage class as the referenced data set.

Specific Volume Requests for System-Managed DASD Data Sets

You can specify one or more volume serial numbers on the VOLUME parameter if the storage administrator has specified GUARANTEED_SPACE=YES in the storage class. In this case, SMS uses the volumes you explicitly specify. If it cannot, the allocation fails. The allocation fails, for example, if not enough space exists on the volumes you specify or if the volumes you specify are not in the list of volumes defined in the storage class, either specified in your JCL or selected by the ACS routines.

For example:

```
//DD14  DD  DSNAME=DESIGNH.PGM,DATACLAS=DCLAS14,STORCLAS=STOR55,
//        DISP=(NEW,KEEP),VOLUME=SER=(223344,334455)
```

If the storage administrator has not specified GUARANTEED_SPACE=YES in the storage class, the system ignores any volume serial numbers you specify for new system-managed DASD data sets.

A system-managed DASD data set can reside on a maximum of 59 volumes.

Nonspecific Volume Requests for System-Managed Data Sets

Omit the VOLUME parameter to make a nonspecific volume request for a new system-managed data set. SMS selects the volume to be used for the data set.

Multivolume Data Sets for System-Managed DASD Data Sets

The system creates a preallocated multivolume data set for system-managed DASD if the storage class has GUARANTEED_SPACE=YES and one of the following:

- The data class has a volume count greater than one
- You specify two or more volume serial numbers
- You specify a volume count greater than one on the VOLUME parameter.

If you choose specific volume serial numbers, the system uses these volumes; otherwise, the system selects the volumes.

Note: All tape volumes in a multivolume data set must reside in the same system-managed tape library and in the same storage group.

Volume Allocation for Non-SMS-Managed Data Sets

Data sets on direct access and magnetic tape reside on or are written on volumes. The volumes may be permanently mounted on the device or may need to be mounted by the operator. To tell the system the volume on which an existing data set resides, make a specific volume request. To tell the system the volume on which to write a new data set, make a specific or nonspecific volume request.

Volume Allocation for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

With SMS, the storage administrator can specify a system default unit. If there is a system default unit, the system uses the volumes associated with the default unit, and you do not need to code the VOLUME parameter.

Volume Attributes

The system assigns volumes two attributes:

- **Use attributes,** which control how volumes are allocated, are:
  - **Private:** The volume can be allocated only when its volume serial number is explicitly or implicitly specified.
  - **Public:** The volume is eligible for allocation to temporary data sets defined with a nonspecific volume request and without a PRIVATE subparameter in the VOLUME parameter.
  - **Storage:** The volume is eligible for allocation to both temporary and permanent data sets defined with a nonspecific volume request and without a PRIVATE subparameter in the VOLUME parameter. Storage volumes usually contain permanent data sets, but can be used for temporary data sets.

- **Mount attributes,** which control how or whether volumes can be demounted after being unallocated, are:
  - **Permanently resident:** The volume, which can only be direct access, cannot be demounted. Volumes that are always permanently resident are all volumes that cannot be physically demounted, the IPL volume, and the volume containing system data sets. Permanently resident volumes have any use attribute.
  - **Reserved:** The volume remains mounted until the operator issues an UNLOAD command. Volumes that should be reserved are volumes that are used frequently by many jobs. Reserved, direct access volumes can have any use attribute; reserved, tape volumes can be only private or public.
  - **Removable:** The volume is neither permanently resident nor reserved. Removable volumes can be demounted after their last use in a job. Removable volumes can be only private or public.

For more information on attributes, see *OS/390 MVS Initialization and Tuning Guide*.

Specific Volume Requests for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

Make a specific volume request by coding:

```
//ddname DD VOLUME=SER=serial-number

//ddname DD VOLUME=REF=dsname
//ddname DD VOLUME=REF=*.ddname

//ddname DD DSNAME=passed data set
//ddname DD DSNAME=cataloged data set
```

For passed or cataloged data sets, the system obtains the volume serial numbers from the passed data set information or from the catalog. In these cases, do not code a SER or REF subparameter in a VOLUME parameter; other VOLUME subparameters can be coded.

How the System Satisfies Specific Volume Requests

In the following cases, the system satisfies a request for a specific volume with a volume that is already mounted:

- The requested volume is permanently resident or reserved. The system assigns the volume regardless of whether public or private use was requested; the volume retains its original use attribute of public or private.
- The requested volume is a removable direct access volume that can be shared and is being used by a concurrently executing step. If the request would make the volume unable to be shared, the system assigns the volume only after all other steps using it terminate.
- The requested volume is a removable direct access volume that is mounted but not allocated. The volume is assigned a use attribute of private if the VOLUME parameter specifies PRIVATE; otherwise, the volume is for public use.
- The requested volume is a scratch tape volume that is mounted but not allocated. The tape is assigned a private attribute if the request is for a

permanent data set or if the VOLUME parameter specifies PRIVATE; otherwise, the volume is for public use.

Note: You cannot make a specific request for a volume that resides in a system-managed tape library and is in the scratch category.

Nonspecific Volume Requests for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

Make a nonspecific volume request for a new data set that can be assigned to any volume or volumes. To make a nonspecific volume request, either:

- Omit the VOLUME parameter.
- Code a VOLUME parameter but omit a SER or REF subparameter.

How the System Satisfies Nonspecific Volume Requests

The system satisfies a request for a nonspecific volume as follows:

Request for private volume for temporary or permanent data set

For removable direct access or tape, the system always asks the operator to mount a volume. The operator should mount a volume containing only unused space so that the owner can control all the space on the volume. Once mounted, the volume is assigned the attribute of private.

For permanently resident direct access, the use of PRIVATE on non-specific requests is not recommended. Operator intervention will be required to allow the system to allocate such a request to a private volume.

Request for public volume for temporary data set

For direct access, the system assigns a public or storage volume that is already mounted or, if no space is available, the system asks the operator to mount a removable volume. If the system selects a mounted, public volume, it remains public. If the operator mounts a volume, it is designated a public volume.

For **non-system-managed tape volumes**, the system assigns any available, mounted, tape volume; if none is available, the system asks the operator to mount a tape volume. Once mounted, the volume is assigned the use attribute of public.

Assigning an available, mounted volume could result in the loss of user data. However, if the tape volumes are labeled and the LABEL parameter specifies the label type, loss of data is usually prevented because the system checks the first record of the tape when opening the data set.

For **system-managed tape volumes**, the system requests that a tape volume be mounted. Once mounted, the volume is assigned the use attribute of public.

Request for public volume for permanent data set

For direct access, the system assigns a storage volume, if one is mounted. Otherwise, the system treats the request as a nonspecific volume request for a private volume, which can be satisfied only by a mountable volume on an available offline device.

For tape volume, the system treats the request as a nonspecific volume request for a private volume.

Private Volumes for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

The system assigns a removable volume a use attribute of private if **any** one of the following is true:

- The VOLUME parameter contains the PRIVATE subparameter.
- The DD statement requests a specific volume.
- The DD statement requests a permanent data set; that is, the data set does not have a system-generated data set name and the DISP parameter does not specify DELETE.

To make a direct access volume private, code:

```
//ddname  DD  VOLUME=PRIVATE
//ddname  DD  VOLUME=SER=xxxxxx
//ddname  DD  VOLUME=REF=*.ddname
//ddname  DD  DSNAME=permanentds,DISP=(,KEEP)
//ddname  DD  DSNAME=permanentds,DISP=(,CATLG)
```

To make a tape volume private, specify or obtain the volume serial number; because the request is for a specific volume, the system automatically makes the tape volume private.

Using Private Volumes

To use a private volume, you must give the system the serial number; the DD statement must specify the serial number or obtain it from the catalog or a from a previous DD statement through a VOLUME=REF parameter.

The system cannot assign a nonspecific volume request to an online permanently-resident or already mounted private volume. Therefore, if you request a private volume, you will be the only one using that volume, unless another job makes a specific volume request for that volume.

Public Volumes for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

The system assigns a removable volume a use attribute of public when **all** of the following are true:

- The VOLUME parameter does **not** contain a PRIVATE subparameter.
- The DD statement does **not** request a specific volume.
- The DD statement requests a temporary data set; that is, no name is specified for the data set name or the disposition is DISP=(NEW,DELETE) or a DISP parameter is omitted to imply a new data set to be deleted.

Volume Affinity for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

Data sets on the same volume have **volume affinity**. Volume affinity influences the allocation of devices. A request for volume affinity with another data set can make the system modify a request for a specific number of units in the unit count subparameter of the UNIT parameter.

“Stacking Data Sets” on page 15-39 provides more information on stacking data sets on the same volume or set of volumes as well as recommendations on which method of volume affinity (explicit versus implicit) you should use.

Explicit Volume Affinity

To request that a new data set be assigned to the same volume(s) as another data set, code:

```
//ddname DD  VOLUME=REF=dsname
//ddname DD  VOLUME=REF=*.ddname
//ddname DD  VOLUME=REF=*.stepname.ddname
//ddname DD  VOLUME=REF=*.stepname.procstepname.ddname
//ddname DD  VOLUME=REF=*.procstepname.ddname
```

Use the first form to reference a cataloged or passed data set. Use the other forms to reference a DD statement earlier in the job.

Implicit Volume Affinity

To request volume affinity implicitly, specify the serial number(s) of the volume(s) containing another data set.

Multivolume Data Sets for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

Number of Volumes

When creating or extending a data set, request the maximum number of volumes that might be required. For non-system-managed data sets, indicate the number in the volume-count specified in the VOLUME parameter, or by the number of serial numbers implicitly or explicitly specified. For datasets on a system-managed tape volume, indicate the number in one of the following ways:

- In the volume-count specified in the VOLUME parameter
- By the number of serial numbers implicitly or explicitly specified
- By specifying a data class that contains the appropriate volume-count definition.

For a multi-volume data set on tape volumes that are:

- System-managed, all volumes must reside in the same system-managed tape library. These volumes must also be part of the same SMS storage group.
- Non-system-managed, all volumes must not be in any system-managed tape library.

If you make a specific volume request for more volumes than units, the system automatically indicates that the volumes allocated to the same unit cannot be shared.

If you request multiple direct access volumes in a JES3 system, they must be either all mountable or all permanently mounted; a mixture is not allowed.

Parallel Mounting

For some jobs, all requested volumes must be mounted before the data set can be used. For these jobs, request as many units as volumes or request parallel mounting by coding P in the UNIT parameter.

Processing Order

When reading or adding to an existing multivolume data set, you can tell the system to begin processing with other than the first volume by coding:

```
//ddname  DD  VOLUME=(,,volume-sequence-number),...
```

Data Sets that Span Libraries

Allocation is able to support volumes created in different tape libraries by treating a single DD statement as though it represents a concatenation of DD statements. In this situation, the volumes must be the same recording technology but can have different media types. When allocation processing encounters a DD statement for an existing multi-volume tape data set whose volumes reside in a tape library, and that DD statement has DCB=OPTCD=B and the volume serial numbers are explicitly coded, allocation processes that statement as though there were additional DD statements, each containing one of the volume serial numbers from the original DD statement. Allocation processing concatenates these DD statements in the order the volume serial numbers were specified on the original DD statement, each having unit affinity with the first DD statement.

For example, when you code:

```
//DD4   DD      DSN=DAYS,DISP=OLD,DCB=OPTCD=B,VOLUME=(SER=793284,227996,382021,
//              427635,946565)
```

allocation processes as though you had coded:

```
//DD4   DD  DSN=DAYS,DISP=OLD,VOLUME=SER=(793284)
//      DD  DSN=DAYS,DISP=OLD,VOLUME=SER=(227996),UNIT=AFF=DD4
//      DD  DSN=DAYS,DISP=OLD,VOLUME=SER=(382021),UNIT=AFF=DD4
//      DD  DSN=DAYS,DISP=OLD,VOLUME=SER=(427635),UNIT=AFF=DD4
//      DD  DSN=DAYS,DISP=OLD,VOLUME=SER=(946565),UNIT=AFF=DD4
```

Note: All of the volumes MUST be in the same storage group.

Volumes Required per DD Statement for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

The maximum number of tape volumes or direct access volumes required to satisfy any DD statement is the greater of:

- volume-count specified in the VOLUME parameter:

  ```
  //ddname  DD  VOLUME=(,,,volume-count),...
  ```

- number of serial numbers implicitly or explicitly specified

The number of serial numbers implicitly or explicitly specified is:

- The number of volume serials in the VOLUME=SER subparameter:

  ```
  //ddname  DD  VOLUME=SER=(serial-number,serial-number,...),...
  ```

- The number of volume serials obtained through VOLUME=REF, if coded:

```
//ddname  DD  VOLUME=REF=dsname
//ddname  DD  VOLUME=REF=*.ddname
```

- The number of volume serials obtained from passed data set information, if the DD statement is receiving a passed data set from a prior step. The receiving DD statement must not specify VOLUME=SER or VOLUME=REF; if it does, the system obtains the number from the VOLUME parameter.
- The number of volume serials obtained from the catalog, if the DD statement requests an existing, cataloged data set. The DD statement must not specify VOLUME=SER or VOLUME=REF; if it does, the system obtains the number from the VOLUME parameter. Also, the data set must not be passed from a prior step.
- The number of volume serials minus the volume sequence number plus one, if the DD statement requests an existing data set and specifies a volume sequence number. For example, if the DD statement specifies eight volume serial numbers and a volume sequence number of four, the system uses five volume serials: 8 - 4 + 1 = 5. The first three volume serials are not used; the first volume that the system allocates is the fourth volume.
- The number of volume serials implied by the unit count in the UNIT parameter, if (1) the unit count is higher than the calculated number of volume serials or (2) the DD statement makes a nonspecific volume request for a new data set on direct access for public use.

When the volume count or unit count require more volumes than the number specified in VOLUME=SER or obtained from VOLUME=REF, passed data set information, or the catalog, the system assumes that the requests are for nonspecific volumes.

Examples

For examples of volume allocation, see “Examples for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume” on page 15-8.

Interactions Between Device and Volume Allocation

Device and volume allocation do not occur in isolation. The device and volume actually allocated to a data set depend on many factors, including such considerations as whether the data set is SMS-managed or non-SMS-managed, whether the data set is cataloged, or whether there is affinity between volumes or data sets. These relationships can be complex. The following sections provide considerations to help you decide how to code these parameters to ensure you are using the resources you want to use.

Relationship of the UNIT and VOLUME Parameters (Non-SMS-Managed Data Sets)

The system can obtain device information from sources other than the UNIT parameter:

- from the catalog for cataloged data sets
- from a passed data set
- from an earlier DD statement
- from another request for the volume in the same step
- system defaults

Cataloged Data Sets

When the data set is cataloged, the system obtains unit and volume information from the catalog. However, if the DD statement for a cataloged data set contains VOLUME=SER=serial-number, the system does not look in the catalog; in this case, you **must** code the UNIT parameter.

Volume References to Cataloged Data Sets

If a data set is to use the same volumes as a cataloged data set, code VOLUME=REF to refer to the cataloged data set. The system obtains unit and volume information from the catalog and places the data set on the same volumes.

Overridden Procedure DD Statements

When a step calls a cataloged or in-stream procedure, an overriding DD statement in the calling step statement can specify a cataloged data set in its DSNAME parameter. If so, the overriding DD statement should nullify the UNIT and VOLUME parameters; if it does not nullify them, the system uses the UNIT and VOLUME parameters on the overridden DD statement and does not search the catalog.

Passed Data Sets

When receiving a data set passed from a previous step, omit the UNIT and VOLUME parameters. The system obtains unit and volume information from the passing step. However, if the receiving DD statement contains VOLUME=SER=serial-number, code the UNIT parameter also.

Earlier DD Statement

If a data set uses the volumes used for a data set in an earlier step, code a VOLUME=REF parameter to refer to the earlier DD statement. The system obtains the unit and volume information from the earlier DD statement. Therefore, you can omit the UNIT parameter. However, to make the system assign more devices or to influence device allocation, code the UNIT parameter. The system uses the coded UNIT parameter, if it requests a subset of the unit type in the referenced DD statement. Otherwise, the system ignores it.

Another Request for the Volume in the Same Step

A volume/unit association may be established during device allocation such that any other request for the volume within the same step will receive the same unit, regardless of the UNIT parameter coded, or the unit default if no UNIT parameter is coded.

In the following example, assume that VOL=SER=NOTSYS is not included in the SYSDA group name, and that the SMS Control Data Set contains a default UNIT of 3380.

```
//DD1   DD  DSN=dsname1,DISP=(CATLG,DELETE),
//          DCB=(DSORG=PS,RECFM=FB,LRECL=80),
//          UNIT=SYSDA,VOL=SER=NOTSYS,SPACE=(80,(1,5 ).RLSE),AVREC=K
/*
//DD2   DD  DSN=dsname2,DISP=(CATLG,DELETE),
//          DCB=(DSORG=PS,RECFM=FB,LRECL=80),
//          VOL=SER=NOTSYS,SPACE=(80,(1,5 ).RLSE),AVREC=K
/*
```

Allocation will initially be unable to allocate DD1 (since NOTSYS is not within SYSDA), so it will temporarily skip it and go on to DD2. For DD2, since no UNIT is specified, Allocation will pick up the default UNIT of 3380, and successfully allocate DD2. It will then go back to DD1, and, recognizing the volume affinity now established with DD2, will ignore the specified UNIT=SYSDA and successfully allocate DD1 to the same 3380 unit.

System Defaults

With SMS, the storage administrator can specify a system default unit. If you create a new data set (specifying DISP=NEW or DISP=MOD) on a system with a system default unit, you can omit the UNIT parameter. SMS supplies the default unit.

There is also a system default for unit affinity processing. This default unit, identified as the unit-affinity-ignored unit name, is specified on UNITAFF in the ALLOCxx PARMLIB member and applies under certain conditions when unit affinity is ignored. See ALLOCxx in *OS/390 MVS Initialization and Tuning Reference* for more information about the default unit-affinity-ignored unit name. Example 5 on page 15-38 shows an example of when this default is used.

It is important to understand how the system uses a group name for the UNIT parameter of a data set that has a disposition of CATALOG or PASS.

- When you specify a group name as the UNIT parameter of a new data set request that you want to catalog, the system stores the generic device type unit information for that data set in the catalog. The system does not retain the group name you originally specify.
- When you specify a group name as the UNIT parameter of a new data set that is to be passed, the system keeps the generic device type unit information for that passed data set. The system does not retain the group name you originally specify.

The following example shows how the system uses unit information it retrieves when it processes subsequent references to a data set that originally specified a group name. Assume this environment for the example:

32 - 3480 tape drives:

- 16 - 3480 tape drives at addresses 3C0-3CF are defined with a group name of TAPE.
- 16 - 3480 tape drives at addresses 4C0-4CF are defined, but are not part of the group named TAPE.

EXAMPLE A

The JCL to create a data set specifies the group name of TAPE as the UNIT parameter:

```
//DD1    DD DSN=A.B,UNIT=TAPE,DISP=(NEW,CATLG)
         or
//DD1    DD DSN=A.B,UNIT=TAPE,DISP=(NEW,PASS)
```

Data set A.B will be allocated to a 3480 tape device that resides at addresses 3C0-3CF.

When a subsequent allocation references data set A.B, the original specification of UNIT=TAPE is no longer available. Subsequent allocations that do not specify a UNIT parameter, such as

```
//DD2    DD DSN=A.B,DISP=SHR
```

will cause data set A.B to be allocated to any 3480 tape drive (that is, addresses 3C0-3CF or 4C0-4CF), because a device type of 3480 is in the catalog.

Note: If the tape containing the data set that was passed from a previous step is still mounted, the system will preferentially leave it on that same drive.

If you desire to have the tape mounted on one of the tape drives defined only to group name TAPE, you must request this by specifying a unit override:

```
//DD2    DD DSN=A.B,DISP=SHR,UNIT=TAPE
```

This will cause the system to consider allocating only the tape drives defined as part of the group name TAPE (3C0-3CF). That is because TAPE is a proper subset of the device information retrieved from the catalog.

Note, however, that if the UNIT parameter specified is not a proper subset of the cataloged (or passed) device type, the system ignores the unit override and allocates the data set to any device matching the retrieved device type information. The following example illustrates this; it shows the effects of unit override when the UNIT parameter does not specify a proper subset of the device information in the catalog (or for a passed data set). In this example, assume the following environment:

32 - 3480 tape drives:

- 16 - 3480 tape drives at addresses 3C0-3CF are defined for the group name of TAPE.
- 16 - 3480 tape drives at addresses 4C0-4CF are defined, but are not part of the group name TAPE.

32 - 3490 tape drives:

- 16 - 3490 tape drives at addresses 3D0-3DF are also defined for the group name of TAPE.
- 16 - 3490 tape drives at addresses 4D0-4DF are defined, but are also not part of the group name TAPE.

EXAMPLE B

The JCL to create a data set specifies a group name of TAPE as the UNIT parameter:

```
//DD1    DD DSN=C.D,UNIT=TAPE,DISP=(NEW,CATLG)
         or
//DD1    DD DSN=C.D,UNIT=TAPE,DISP=(NEW,PASS)
```

Data set C.D will be allocated to a 3480 or 3490 device that resides at addresses 3C0-3CF or 3D0-3DF.

When a subsequent allocation references data set C.D, the original specification of UNIT=TAPE is no longer available. Subsequent allocations that do not specify a UNIT parameter, such as

```
//DD2    DD DSN=C.D,DISP=SHR
```

will cause data set C.D to be allocated to any 3480 tape drive (that is, 3C0-3CF or 4C0-4CF) if the data set was originally created on a 3480, or to any 3490 tape drive (that is, 3D0-3DF or 4D0-4DF) if the data set was originally created on a 3490.

Note: If the tape containing the data set that was passed from a previous step is still mounted, the system will preferentially leave it on that same drive.

It is not possible using unit overrides to specify that the tape be mounted on one of the tape drives defined only to the group named TAPE. This is because the retrieved device type information will have only one device type (3480 or 3490), whereas two device types (3480 and 3490) are defined to the group named TAPE, so TAPE is not a proper subset of the one device type that is retrieved.

In Example B, a unit override of TAPE will be ignored and the data set on the DD2 statement will be allocated to any device matching the cataloged (or passed) device type. That is, if the cataloged (or passed) device type was 3480, the data set will be allocated to 3C0-3CF or 4C0-4CF; if the cataloged (or passed) device type was 3490, the data set will be allocated to 3D0-3DF or 4D0-4DF.

Note: It is possible for installations to influence device selection for non-SMS-managed tape requests through the Tape Device Selection call (SSI function code 78). See "Using the Subsystem Interface" for more information.

Relationship of the UNIT and VOLUME Parameters (SMS-Managed Data Sets)

The system can obtain device eligibility information from:

- the catalog for cataloged tape data sets
- a passed data set
- an earlier DD statement
- a previous request for the volume
- the tape configuration database

Cataloged Data Sets

When a tape data set is cataloged, the system obtains device eligibility and volume information from the catalog. If the DD statement for a cataloged tape data set contains a volume serial number that is not in the SMS configuration, the system does not use the catalog; instead, it obtains device eligibility from the tape configuration data base.

For data sets with no volume serial specified, the system always searches the catalog when a data set is OLD (or MOD treated as OLD). For data sets with a non-SMS volume serial specified, the system assumes the data set is non-SMS-managed and resides on that volume, and it does not search the catalog. For data sets with an SMS volume serial number specified (whether it is a real volume or a non-existent volume that is in a DUMMY storage group), the system always searches the catalog; SMS controls volume selection, and the data set might not be on the volume that is specified.

Volume References to Cataloged Data Sets

For tape, if a data set is to use the same volume(s) as a cataloged data set, code VOLUME=REF to refer to the cataloged data set. The system obtains unit and volume information from the catalog and places the data set on the same volume(s). For DASD, storage class and volume information are retrieved from the catalog. The data sets share the same storage class but can be in different storage groups as long as the storage groups are of compatible types (such as POOL or VIO).

Overridden Procedure DD Statements

When a step calls a cataloged or in-stream procedure, an overriding DD statement in the calling step statement can specify a cataloged data set in its DSNAME parameter. If so, the overriding DD statement should nullify the VOLUME parameter; otherwise, the system uses the VOLUME parameter on the overridden DD statement and does not search the catalog.

Passed Data Sets

When receiving a data set passed from a previous step, omit the VOLUME parameter. The system obtains volume information from the passing step.

Earlier DD Statement

For NEW data sets, if VOL=REF references a non-SMS-managed data set, the ACS routines are given control. The ACS routines can either allow the non-SMS

allocation or fail it, but they cannot make the new data set SMS-managed. For NEW data sets, if VOL=REF references an SMS-managed data set, then:

- For tape, the storage group and the volume must be the same as the referenced data set.
- For DASD, the storage class must be the same as the referenced data set, but the storage groups can be different as long as they are compatible, such as POOL and VIO.

For OLD data sets, if VOL=REF references a non-SMS-managed data set, the system assumes the data set is non-SMS-managed and on the same volume as the referenced data set. If VOL=REF references an SMS-managed data set, the system searches the catalog because the referencing data set might not be on the same volume as the referenced data set.

Previous Request for the Volume

A volume/unit association can be established during device allocation so that any subsequent request for the volume within the same step will receive the same unit.

Unit and Volume Affinity for Non-System-Managed Data Sets and Data Sets on a System-Managed Tape Volume

When two or more volumes are assigned the same device, the volumes are said to have **unit affinity** within the same job step allocation. Unit affinity implies deferred mounting for all except one of the volumes.

The following definitions apply to unit affinity:

- A **referencing DD** is the DD that specifies the UNIT=AFF keyword.
- A **referenced DD** is the DD pointed to by a referencing DD.
- A **primary DD** is the first DD in a unit affinity chain.
- A **unit affinity chain** is the set of DDs that share the same primary DD.

In the following example, DD2 is a referencing DD; DD1 is its referenced DD. DD3 is also a referencing DD; DD2 is its referenced DD. DD1 is the primary DD for the unit affinity chain that consists of the DD1, DD2, and DD3.

```
//ST1 EXEC
//DD1 DD DSN=A,DISP=(,CATLG),UNIT=3480
//DD2 DD DSN=B,DISP=(,CATLG),UNIT=AFF=DD1
//DD3 DD DSN=C,DISP=(,CATLG),UNIT=AFF=DD2
```

A related concept is volume affinity. When two or more data sets share one or more volumes, the data sets have **volume affinity**. See “Stacking Data Sets” on page 15-39 for additional information on stacking data sets on one or more volumes.

Explicit Unit Affinity

To reduce the number of devices for a step, code UNIT=AFF to request that an existing data set be assigned to the same device(s) assigned for an earlier DD statement in the same step. Code:

```
//ddname DD  UNIT=AFF=ddname,...
```

Note: Do not specify UNIT=AFF for a NEW (or MOD treated as NEW) data set that references a non-SMS-managed DASD data set; the allocation will fail.

For concatenated data sets, code the following to assign the data sets to the same device:

```
//DD1    DD  DSNAME=dataset1,...
//       DD  DSNAME=dataset2,UNIT=AFF=DD1,...
//       DD  DSNAME=dataset3,UNIT=AFF=DD1,...
```

When you use explicit unit affinity, it is recommended that you use UNIT=AFF to reference the previous DD in the unit affinity chain, rather than the primary DD. That is, code:

```
//DD1    DD  DSNAME=dataset1,...
//DD2    DD  DSNAME=dataset2,UNIT=AFF=DD1,...
//DD3    DD  DSNAME=dataset3,UNIT=AFF=DD2,...
//DD4    DD  DSNAME=dataset3,UNIT=AFF=DD3,...
```

rather than:

```
//DD1    DD  DSNAME=dataset1,...
//DD2    DD  DSNAME=dataset2,UNIT=AFF=DD1,...
//DD3    DD  DSNAME=dataset3,UNIT=AFF=DD1,...
//DD4    DD  DSNAME=dataset3,UNIT=AFF=DD1,...
```

Always referencing the previous DD means that, if any condition causes the system to ignore unit affinity for one of the DDs in the chain, any subsequent DDs in the chain will still be allocated to a single unit, rather than to different units.

Implied Unit Affinity

Implied unit affinity exists among the volumes for one data set when the DD statement requests more volumes than devices.

Attention: If all of the following conditions are present, the data set on the referencing DD statement, which requests unit affinity, is written over by the data set on the referenced DD statement:

- The referenced DD statement makes a nonspecific volume request.
- The data set on the referencing DD statement is opened before the referenced data set.
- The tape is not unloaded before the referenced data set is opened and the LABEL parameter does not request positioning of the tape to check tape labels. A tape device allocated to more than one data set is not unloaded when it is dynamically unallocated, or when it is closed and FREE=CLOSE is specified.

Unit Affinity Processing for Data Sets on a System-Managed Tape Volume

Figure 15-6 on page 15-32 contains examples that apply unit-affinity principles to data sets requested on system-managed tape volumes. The system verifies that the primary (referenced) DD statement has a device pool that is a proper subset of the secondary (referencing) DD statement. Therefore, the system honors unit-affinity requests only when each device type to which the primary DD statement is eligible is also contained in the device pool of the secondary DD statement.

The column headings have the following meanings:

Request Indicates either primary (referenced) DD statement or secondary (referencing) DD statement.

Library Eligibility
Indicates the libraries to which the DD statement is eligible.

Device Type Eligibility
Indicates the device types to which the DD statement is eligible.

Action Taken
Whether the affinity request is honored or ignored. If the request is honored, whether the library or device type eligibility is reduced.

Final Eligibility
The resulting library or device type eligibility used to allocate devices.

Figure 15-6 (Page 1 of 2). Unit-Affinity Examples of Tape Library Requests

Requestor	Library Eligibility	Device Type Eligibility	Action Taken	Final Eligibility
Libraries and device pools of requestors are identical				
Primary	LIB1	3490	Honor	LIB1, 3490
Secondary	LIB1	3490		
Libraries of primary requestor are proper subset of secondary				
Primary	LIB1	3490	Honor and Reduce	LIB1, 3490
Secondary	LIB1, LIB2	3490		
Device pool of primary requestor is proper subset of secondary				
Primary	LIB1	3490	Honor and Reduce	LIB1 and 3490
Secondary	LIB1	3490, 348X		
Libraries of primary requestor are completely different from secondary				
Primary	LIB1	3490	Ignore	LIB1, 3490
Secondary	LIB2	3490		LIB2, 3490
Device pool of primary requestor is completely different from secondary				
Primary	LIB1	3490	Ignore	LIB1, 3490
Secondary	LIB1	348X		LIB1, 348X
Libraries of primary requestor are not proper subset of secondary				
Primary	LIB1, LIB2	3490	Ignore	LIB1, LIB2, 3490
Secondary	LIB1	3490		LIB1, 3490
Device pool of primary requestor is not proper subset of secondary				
Primary	LIB1	3490, 348X	Ignore	LIB1, 3490, 348X
Secondary	LIB1	348X		LIB1, 348X
Device pools identical; both are non-library requestors				
Primary	Non-library request	3490	Honor	3490
Secondary	Non-library request	3490		
Device pool of primary requestor is proper subset of secondary; both are non-library requestors				

Figure 15-6 (Page 2 of 2). Unit-Affinity Examples of Tape Library Requests

Requestor	Library Eligibility	Device Type Eligibility	Action Taken	Final Eligibility
Primary	Non-library request	3490	Honor and Reduce	3490
Secondary	Non-library request	3490, 348X		
Device pool of primary requestor is not proper subset of secondary; both are non-library requestors				
Primary	Non-library request	3490, 348X	Ignore	3490, 348X
Secondary	Non-library request	3490		3490
Primary is library requestor but secondary is non-library requestor				
Primary	LIB1	3490	Ignore	LIB1, 3490
Secondary	Non-library request	3490		3490

Note: 348X is the device type for the 3490 model tape drives and 3490 is the device type for the 3490E model tape drives.

Device Eligibility

Non-system-managed data sets are eligible to a device when they can be allocated to that device type. The datasets on a system-managed tape volume are eligible to a device when they can be allocated to that device type, and when both the volume and the device reside in the same system-managed tape library. The catalog contains information about the types of devices to which a data set is eligible only if the data set is cataloged.

For the system to honor a request for unit affinity, the referenced DD must be eligible to the same devices as the referencing DD statement. In addition, the devices to which the referenced DD statement is eligible must either be a subset of, or the same as, the devices to which the referencing DD is eligible. In all other cases, the system ignores unit affinity, but the allocation will succeed.

These rules are illustrated by the following example, in which:

- TAPEX is eligible to a 3480X or a 3480.
- DS3480 is cataloged as eligible to a 3480. The unit name 3480 has two generic names associated with it: 3480 and 3480X.
- DS3480X is cataloged as eligible to a 3480X.
- DS3480X2 is cataloged as eligible to a 3480X.

```
//DD1 DD UNIT=TAPEX
//DD2 DD DSN=DS3480,UNIT=AFF=DD1
//DD3 DD DSN=DS3480X,UNIT=AFF=DD2
//DD4 DD DSN=DS3480X2,UNIT=AFF=DD1
```

If you do not request volume affinity, or the request for volume affinity does not break the unit affinity (see “Interaction of Unit and Volume Affinity Requests” on page 15-34), the following unit affinities will result:

- DD1 and DD2 can have unit affinity, because DD1 and DD2 are both eligible to a 3480 and a 3480X.
- DD4 can have unit affinity to DD3, because DD3 and DD4 are both eligible to a 3480X.

- Neither DD3 nor DD4 can have unit affinity to DD1, because neither is eligible to a 3480. Thus, the system ignores unit affinity for DD3 or DD4; DD3 and DD4 are not eligible for the same devices as DD1.

Exception to Device Eligibility

The system will **not** honor unit affinity when all of the following conditions are met:

- The referenced DD is eligible to a 3480X device
- The referencing DD is eligible to both a 3480 and a 3480X device
- The system was initialized to attempt to allocate a 3480 before a 3480X.

The exception is illustrated by the following example, in which:

- The system has been initialized to attempt to allocate a 3480 device before allocating a 3480X device.
- DS3480 is cataloged as eligible to a 3480. The unit name 3480 has two generic names associated with it: 3480 and 3480X.

```
//DD1 DD UNIT=3480X
//DD2 DD DSN=DS3480,UNIT=AFF=DD1
```

In this example, the system does not honor the request for unit affinity; each DD statement is allocated to a separate device.

Interaction of Unit and Volume Affinity Requests

Unit affinity, volume affinity, and/or unit and volume affinity can exist in the same step and on the same DD statement.

If both unit and volume affinity are requested in the same step, sometimes only one affinity can be honored. Figure 15-7 indicates how the system honors unit and volume affinity requests for either tape or direct access devices.

Figure 15-7 (Page 1 of 2). Unit and Volume Affinity (Non-SMS-Managed Data Sets)

Relationship of Unit and Volume Affinity Requests	**Tape**	**Direct Access**
All unit and volume affinity requests unrelated	The system honors all unit and volume affinity requests.	

Example for Tape:
//DD1 DD VOLUME=SER=A,UNIT=3490
//DD2 DD VOLUME=SER=B,UNIT=AFF=DD1
//DD3 DD VOLUME=SER=(C,D),UNIT=3490
//DD4 DD VOLUME=SER=C,UNIT=3490
	The system assigns DD2 to the same unit as DD1. The system uses the same unit for volume C for both DD3 and DD4. The system will allocate a total of 3 devices for this series of requests.	

Example for Direct Access:
//DD1 DD VOLUME=SER=A,UNIT=3340
//DD2 DD VOLUME=SER=B,UNIT=AFF=DD1
//DD3 DD VOLUME=SER=C,UNIT=3340
//DD4 DD VOLUME=SER=C,UNIT=3340
1. Unit affinity is explicitly requested between DD1 and DD2.
2. Volume affinity is implicitly requested between DD3 and DD4.
		The system assigns DD2 to the same unit as DD1. The system uses the same unit for volume C for both DD3 and DD4. The system will allocate a total of 2 devices for this series of requests.

Figure 15-7 (Page 2 of 2). Unit and Volume Affinity (Non-SMS-Managed Data Sets)

<table>
<tr><th>Relationship of Unit and Volume Affinity Requests</th><th>Tape</th><th>Direct Access</th></tr>
<tr><td rowspan="3">All unit and volume affinity requests related

Example for Tape:
//DD1 DD VOLUME=SER=(A,D),UNIT=3490
//DD2 DD VOLUME=SER=(A,B),
// UNIT=AFF=DD1

Example for Direct Access:
//DD1 DD VOLUME=SER=(A,D),UNIT=3340
//DD2 DD VOLUME=SER=(A,B),
// UNIT=AFF=DD1
1. DD1 implies unit affinity because both volumes use the same unit.
2. Unit affinity is explicitly requested between DD1 and DD2.
3. Volume affinity is implicitly requested between DD1 and DD2, because both request volume A.</td><td>The system honors all unit affinity requests and ignores all volume affinity requests. Results: all volumes use the same unit.</td><td>The system honors all volume affinities contained in the unit affinity request; these volumes use the same unit. The other volumes in the unit affinity request use a different unit.</td></tr>
<tr><td>The system assigns DD2 to the same unit as DD1.</td><td></td></tr>
<tr><td></td><td>The system assigns volume A for DD2 to the same 3340 as volume A for DD1. Volumes D and B use the other 3340.</td></tr>
<tr><td rowspan="3">Some unit and volume affinities related, some unrelated

Example for Tape:
//DD1 DD VOLUME=SER=A,UNIT=3490
//DD2 DD VOLUME=SER=B,UNIT=AFF=DD1
//DD3 DD VOLUME=SER=B,UNIT=AFF=DD2

Example for Direct Access:
//DD1 DD VOLUME=SER=A,UNIT=3340
//DD2 DD VOLUME=SER=B,UNIT=AFF=DD1
//DD3 DD VOLUME=SER=B,UNIT=3340
1. Unit affinity is explicitly requested between DD1 and DD2.
2. Volume affinity is implicitly requested between DD2 and DD3.</td><td colspan="2">The system honors all volume affinities contained in the unit affinity request; these volumes use the same unit. The other volumes in the unit affinity request use a different unit.</td></tr>
<tr><td>The system assigns DD2 to the same unit as DD1. Volume B (in DD2 and DD3) is a 3490 volume. Thus, DD1, DD2, and DD3 use one 3490.</td><td></td></tr>
<tr><td></td><td>The system assigns volume B for DD2 and DD3 to one 3340. Volume A for DD1 uses another 3340.</td></tr>
</table>

Permanently Resident or Reserved Volumes

If a DD statement requests a volume that is a permanently-resident or reserved volume, the system must allocate the device on which the volume is mounted, regardless of any affinities requested.

UNIT=AFF when Requesting Extended Data Sets in a JES3 System

In a multiple-step job in a JES3 system, if a data set is extended in an early job step to additional volumes, MVS allocates the additional devices needed. JES3 is unaware of the additional devices. If a later step requests the data set, code UNIT=AFF=ddname so that the system allocates the original and additional devices for the data set.

Affinity for Multivolume Data Sets

For multivolume data sets, request volume affinity if you request unit affinity. Code:

```
//ddname  DD  UNIT=AFF=ddname,VOLUME=REF=*.ddname,...
```

If you code only volume affinity for a multivolume data set, the following can happen:

- The system assigns the requested volumes and allocates them to a device. Thus, the device is to be shared by all the DD statements requesting volume affinity.
- The system asks the operator to mount the first volume for the referenced DD statement on the allocated device.
- At the end of the first volume, the system asks the operator to demount the first volume and mount the second volume.
- If the data set is reopened, the system asks the operator to remount the first volume on a device not used for the volume affinity request.
- When the system processes the referring DD statement, it asks the operator to mount the first volume on the device assigned to the volume affinity request. The job now enters a wait because the system has requested the first volume on two different devices.

Device Usage for Data Sets on a System-Managed Tape Volume

Unit affinity requests are honored if each tape volume associated with the DD statements resides in the same system-managed tape library. Otherwise, such volumes cannot share the same device, and UNIT=AFF (unit affinity) requests are ignored. For example:

```
//STEP 1 EXEC PGM=...
//DD1 DD DSN=SAM,VOL=SER=TAPEA
//DD2 DD DSN=SAM,VOL=SER=TAPEB,UNIT=AFF=DD1
```

If neither volume TAPEA nor volume TAPEB resides in a system-managed tape library or if both TAPEA and TAPEB reside in the same system-managed tape library, then DD1 and DD2 will share one device; only one device is allocated to job step STEP1. Otherwise, DD1 and DD2 require separate devices; two devices are allocated to job step STEP1.

To control the number of devices allocated, consider the relationship of DD statements and volumes **before** moving existing volumes into a system-managed tape library and when choosing a system-managed tape library to create data sets that will be referenced in a UNIT=AFF statement. DD statements that specify unit affinity might require more devices after associated volumes are moved into a system-managed tape library.

Examples of When the System Ignores Unit Affinity

The following examples show cases when the system ignores the request for unit affinity but allocates the data sets. These are cases where the user specified the UNIT=AFF keyword to limit the number of devices the job requires, but the volumes required are on different device types and thus require separate units.

For example, if your installation uses tape mount management (TMM) methodology, it is possible the ACS routines will redirect some, but not necessarily all, of the DDs in a unit affinity chain to SMS-managed DASD. This redirection can cause a mix of

different device categories (such as SMS-managed tape, SMS-managed DASD, non-SMS-managed tape, non-SMS-managed DASD) within a unit affinity chain, as shown in Examples 1 and 5.

When the system ignores unit affinity, message IEF278I indicates that unit affinity was ignored and provides a reason code.

Example 1

```
//DD1 DD DSNAME=P,DISP=NEW,UNIT=3480
//*               (P is redirected to SMSD, an SMS-managed DASD volume)
//DD2 DD DSNAME=Q,DISP=NEW,UNIT=AFF=DD1
//*               (Q is redirected to SMST, an SMS-managed TAPE volume)
```

In this example, the ACS routines have redirected data set P from non-SMS-managed tape to SMSD, an SMS-managed DASD volume; the ACS routines have also redirected data set Q from non-SMS-managed tape to SMST, an SMS-managed tape volume. DD2 requests unit affinity with DD1, but the system ignores the request because the redirection resulted in inconsistent device categories.

The system issues message IEF278I with reason code 1, indicating that one of the DDs is an SMS-managed request and the other is not.

Example 2

```
//DD1 DD DSNAME=PAYROLL,DISP=OLD
```

PAYROLL is a generation data group (GDG). DD1 is a GDG ALL request. The system treats a GDG ALL request like a concatenation of all the PAYROLL data sets in the catalog (most recent first). All subsequent generations have unit affinity to the first.

The following example shows the JCL the system creates for the GDG ALL request for the PAYROLL data set; the catalog contains 4 entries, one on tape and three on DASD.

```
//DD1 DD DSNAME=PAYROLL(0),DISP=OLD,UNIT=3480,
//       VOLUME=SER=TAPE01
//    DD DSNAME=PAYROLL(-1),DISP=OLD
//       VOLUME=SER=DISK03,UNIT=AFF=DD1
//    DD DSNAME=PAYROLL(-2),DISP=OLD,
//       VOLUME=SER=DISK02,UNIT=AFF=DD1
//    DD DSNAME=PAYROLL(-3),DISP=OLD,
//       VOLUME=SER=DISK01,UNIT=AFF=DD1
```

The system ignores unit affinity. PAYROLL(0) is a tape and cannot share a unit with the other data sets, which reside on DASD. Because the DASD volumes are non-removable, the system allocates a separate volume to PAYROLL(-1), to PAYROLL(-2), and to PAYROLL(-3).

The system issues message IEF278I with a reason code of 2, indicating that the DDs requested incompatible generics.

Example 3

```
//DD1 DD DSNAME=P,DISP=OLD
//*           (P is cataloged on TEST1 in tape Library TL1)
//*           (Tape Library TL1 is eligible to 3480 devices)
//DD2 DD DSNAME=Q,DISP=OLD,UNIT=AFF=DD1
//*           (Q is cataloged on TEST2 in tape Library TL2)
//*           (Tape Library TL2 is eligible to 3490 devices)
```

The system ignores the unit affinity request. P is cataloged on volume TEST1, which resides in the TL1 tape library, and Q is cataloged on volume TEST2, which resides in the TL2 tape library.

The system issues IEF278I with a reason code of 3, indicating that the DDs requested incompatible tape libraries.

Example 4

```
//DD1 DD DSNAME=R,DISP=OLD
//*                  (R is cataloged on T2, a 3480 tape)
//DD2 DD DSNAME=S,DISP=OLD,UNIT=AFF=DD1
//*                  (S is cataloged on T3, a 3480X tape)
```

The system ignores the unit affinity request. DD1 is a 3480 tape volume, but DD2 needs a 3480X tape volume, which is not compatible with 3480.

The system issues message IEF278I with a reason code of 4, indicating that devices associated with the referenced DD (DD1) are not a subset of the devices associated with the referencing DD (DD2).

Example 5

```
//S1  EXEC ...
//DD1 DD DSNAME=W,DISP=(,CATALG),UNIT=3480,VOL=SER=TAPE01
//*            (W is redirected to SD2, an SMS-managed DASD volume)
//S2  EXEC ...
//DD1 DD DSNAME=W,DISP=OLD
//*            (W is cataloged on SD2, an SMS-managed DASD volume)
//DD2 DD DSNAME=X,DISP=NEW,UNIT=AFF=DD1
//*            (X is non-SMS-managed after ACS routine processing)
```

In this example, the ACS routines have redirected data set W from non-SMS-managed tape to SD2, an SMS-managed DASD volume; the ACS routines have not redirected data set X. The system cannot honor the unit affinity request for DD2 in step S2 because the redirection resulted in inconsistent device categories. Therefore, the system allocates data set X as a non-SMS-managed data set on the default unit-affinity-ignored unit (named on UNITAFF in the ALLOCxx parmlib member).

The system issues message IEF278I with a reason code of 5, indicating that the referencing request (DD2) is a non-SMS-managed data set and the referenced request (DD1) is an SMS-managed data set.

Stacking Data Sets

When two or more data sets are placed on the same tape volume or set of tape volumes, the data sets are said to be stacked. Use **data set stacking** to increase the efficiency of tape media usage and to decrease the number of tape volumes needed by allocation. Data set stacking is also useful if you need to send data offsite; you can group related data sets together on a minimum number of tape volumes.

A **data set collection** is the collection of data sets that are intended to be allocated on the same tape volume or set of tape volumes as a result of data set stacking. Data sets can be stacked on a single volume (that is, a data set resides on one volume but shares that volume with at least one other data set). Data sets can also be stacked on multiple volumes (that is, a data set spans two or more volumes and shares at least one of those volumes with one or more data sets or portions of data sets).

You request data set stacking specifying the data set sequence number on the LABEL parameter in combination with either the volume reference (VOL=REF) or volume serial (VOL=SER) parameters. You can request data set stacking within the same step, across steps within the same job, or across jobs.

Use the following table to determine the JCL parameters needed to request data set stacking. This table shows which parameter (VOL=SER or VOL=REF) IBM recommends that you use when you want to request data set stacking. For example, to request that multiple data sets in different steps of the job be stacked on the same tape volume, you need to specify VOL=REF by data set name to the previous data set placed on the tape.

Figure 15-8. IBM-Recommended Parameters for Data Set Stacking

Volume Set	Same Step	Different Step, Same Job	Different Job
Single Volume, Multiple Data Sets	VOL=REF to previous data set on the tape	VOL=REF by data set name to previous data set	VOL=REF by data set name to previous data set
Multiple Volumes, Multiple Data Sets	VOL=SER with last volume serial of the previous data set as the first volume serial in the list	VOL=REF by data set name to previous data set	VOL=REF by data set name to previous data set

The following JCL shows an example of data set stacking:

```
//      JOB ...
//S1    EXEC ...
//D1    DD    DSN=A,DISP=(NEW,CATLG),VOL=SER=VOL1,
//            UNIT=TAPE,LABEL=(1,SL)
//D2    DD    DSN=B,DISP=(NEW,CATLG),VOL=REF=*.D1
//            UNIT=TAPE,LABEL=(2,SL)
```

This JCL stacks two data sets on a single volume within the same step. In this example, VOL=REF is used to stack both data set A and data set B on the same tape volume, VOL1. Data sets A and B make up the data set collection.

Examples of Data Set Stacking

The following additional JCL examples request data set stacking; these examples follow the IBM recommendations for specifying data set stacking.

Example 1

This example shows stacking multiple data sets on a single volume within the same step.

```
//ST1 EXEC ...
//DD1 DD DSNAME=W,DISP=OLD  (where W is on volume SMST)
//DD2 DD DSNAME=X,DISP=NEW,VOLUME=REF=*.DD1,
//       LABEL=(2,SL)
//DD3 DD DSNAME=Y,DISP=NEW,VOLUME=REF=*.DD1,
//       LABEL=(3,SL)
```

In this example, VOL=REF is used to stack data sets W, X, and Y on the same tape volume, SMST. Data sets W, X, and Y make up the data set collection.

Example 2

This example shows stacking multiple data sets on a single volume across steps within a job.

```
//JOB1 JOB ...
//ST1 EXEC ...
//DD1 DD DSN=A,DISP=(NEW,CATLG),UNIT=TAPE
//ST2 EXEC ...
//DD2 DD DSN=B,DISP=NEW,LABEL=(2,SL),
//    VOLUME=REF=A,UNIT=TAPE
```

In this example, VOL=REF by data set name is used to stack data sets A and B on the same tape volume. Because DD1 does not specify VOL=SER, DD1 represents a non-specific tape request, so the system assigns an available tape volume or asks the operator to mount a tape volume, if none is available. DD2 places data set B on the same volume as data set A. Data sets A and B make up the data set collection.

Example 3

This example shows stacking multiple data sets on a single volume across jobs.

```
//JOB1 JOB ...
//ST1 EXEC ...
//DD1 DD DSN=A,DISP=(NEW,CATLG)

//JOB2 JOB ...
//ST2 EXEC ...
//DD2  DD DSN=B,DISP=NEW,LABEL=(2,SL),
//     VOLUME=REF=A,UNIT=TAPE
```

In this example, VOL=REF by data set name is used to stack data sets A and B on the same tape volume. Because DD1 on JOB1 does not specify VOL=SER, DD1 represents a non-specific tape request, so the system assigns an available tape volume or asks the operator to mount a tape volume, if none is available. DD2 places data set B on the same volume as data set A. Data sets A and B make up the data set collection.

Example 4

This example shows stacking multiple data sets on multiple volumes within the same step.

```
//ST1 EXEC ...
//DD1 DD DSNAME=W,DISP=NEW,
//       VOLUME=SER=(ONE,TWO,THREE)
//DD2 DD DSNAME=X,DISP=NEW,
//       VOLUME=SER=(THREE,FOUR),
//       LABEL=(2,SL)
//DD3 DD DSNAME=Y,DISP=NEW,VOLUME=SER=(FOUR,FIVE),
//       LABEL=(3,SL)
```

In this example, specifying VOL=SER to refer to the last volume of the previous DD is used to stack data sets W, X, and Y on the same set of tape volumes. Data sets W, X, and Y make up the data set collection.

Example 5

This example shows stacking multiple data sets on a multiple volumes within the same step. Data set W is an existing, multivolume data set on volumes V1 and V2.

```
//ST1 EXEC ...
//DD1 DD DSNAME=W,DISP=OLD (W is on volumes V1 and V2)
//DD2 DD DSNAME=X,DISP=NEW,
//       VOLUME=SER=(V2,V3),
//       LABEL=(2,SL)
//DD3 DD DSNAME=Y,DISP=NEW,
//       VOLUME=SER=(V3,V4),
//       LABEL=(3,SL)
```

In this example, specifying VOL=SER to refer to the last volume of the previous DD is used to stack data sets W, X, and Y on the same tape volumes. Data sets W, X, and Y make up the data set collection.

Example 6

This example shows stacking multiple data sets on multiple volumes across steps in the same job.

```
//JOB1 JOB ...
//ST1 EXEC ...
//DD1 DD DSN=A,DISP=(NEW,CATLG),UNIT=TAPE,
//       VOLUME=SER=(ONE,TWO,THREE)
//ST2 EXEC ...
//DD2 DD DSN=B,DISP=NEW,LABEL=(2,SL),
//    VOLUME=REF=A,UNIT=TAPE
```

In this example, specifying VOL=REF by data set name is used to stack data sets A and B on the same tape volume, THREE. Data sets A and B make up the data set collection.

Example 7

This example shows stacking multiple data sets on multiple volumes across jobs.

```
//JOB1 JOB ...
//ST1 EXEC ...
//DD1 DD DSN=A,DISP=(NEW,CATLG),UNIT=TAPE,
//       VOLUME=SER=(ONE,TWO,THREE))

//JOB2 JOB ...
//ST2 EXEC ...
//DD2 DD DSN=B,DISP=NEW,LABEL=(2,SL),
//       VOLUME=REF=A,UNIT=TAPE
```

In this example, specifying VOL=REF by data set name is used to stack data sets A and B on the same tape volume (THREE, in this case). Data sets A and B make up the data set collection.

Data Set Stacking and TMM

If you are a system programmer or storage administrator and your installation plans to take advantage of tape mount management (TMM) methodology, you need to understand its effect on existing practices. Using TMM can improve the effectiveness of tape device usage because you can redirect certain types of tape data sets to DASD.

Based on your analysis of the output from the Volume Mount Analyzer, you might identify data sets that would appear to be good candidates for redirection from tape to DASD. If, however, your installation has jobs that use data set stacking, you need to make sure that either all of the data sets in a data set collection are redirected or that none of the data sets in a data set collection are redirected. Otherwise, there might be more than one device category for the data sets in the collection, a problem that could cause allocation or open failures.

The term **device category** refers to types of devices. The categories are:

- SMS DASD
- SMS Tape
- Non-SMS-managed DASD
- Non-SMS-managed tape

You can request data set stacking with either VOL=SER or VOL=REF. With VOL=SER, the system can detect data set stacking and check for consistent device categories only within a step. To request data set stacking across steps or across jobs, you must use VOL=REF.

When you specify VOL=SER to request data set stacking within a step, the system checks for mixed device categories. If the system finds mixed device categories within a data set collection, it invokes the ACS routines to try to resolve the device category conflict. If the ACS routines do not direct the data sets to a consistent device category, the allocation fails with message IGD23101I. **Note:** The system does not include existing SMS-managed data sets in a data set collection because catalog information might reflect a redirection. See Example 3.

DFSMS/MVS Implementing System-Managed Storage provides more information about ACS routine handling and detection of data set stacking.

When you specify VOL=REF to request data set stacking across steps or jobs, the system can pass information about the volume references to the ACS routines.

With this information, the ACS routines can direct requests for data sets within a data set collection to the same device category.

If your installation is using TMM and runs jobs that request data set stacking, you need to understand that, because ACS routines might redirect data sets from tape to DASD, certain JCL combinations might not produce the results you expect. Thus:

- IBM recommends that you use VOL=REF to request data set stacking across steps or jobs

While you might find that specifying VOL=SER to request data set stacking across steps or jobs does work sometimes, it might not always produce the results you expect. To avoid problems, use VOL=REF.

If you are using or planning to use TMM and want to use data set stacking, eliminate requests for data set stacking like the ones shown in the following examples. (For examples that show recommended methods of requesting data set stacking, see “Examples of Data Set Stacking” on page 15-40.)

Example 1

```
//JOB1 JOB .....
//STEP1 EXEC .....
//DD1 DD DSNAME=W,DISP=(NEW,CATLG),
//       VOLUME=SER=MINE,UNIT=3490
//STEP2 EXEC .....
//DD2 DD DSNAME=X,DISP=NEW,VOLUME=SER=MINE,
//       LABEL=(2,SL),UNIT=3490
```

This example uses VOL=SER to request data set stacking across steps.

If you replace VOL=SER in DD2 with VOL=REF=W, the ACS routines will have the information they need to allocate data set X to a consistent device category even if data set W is redirected to DASD.

Example 2

```
//JOB1 JOB .....
//DD1 DD DSNAME=W,DISP=(NEW,CATLG),
//       VOLUME=SER=MINE,UNIT=3490

//JOB2 JOB .....
//DD2 DD DSNAME=X,DISP=NEW,VOLUME=SER=MINE,
//       LABEL=(2,SL),UNIT=3490
```

This example uses VOL=SER to request data set stacking across jobs.

If you replace VOL=SER in DD2 with VOL=REF=W, the ACS routines will have the information they need to allocate data set X to a consistent device category.

Example 3

This example uses VOL=SER to request data set stacking across jobs.

```
//JOB1 JOB .....
//STEP1 EXEC .....
//DD1 DD DSNAME=W,DISP=(NEW,CATLG),UNIT=3490,
//       VOL=SER=TAPE01,LABEL=(1,SL)
```

In JOB1, the ACS routines redirect data set W to SMS-managed DASD. Data set W becomes SMS-managed.

```
//JOB2 JOB .....
//STEP2 EXEC .....
//DD1 DD DSNAME=W,DISP=OLD
//DD2 DD DSNAME=X,DISP=NEW,VOL=SER=TAPE01,
//       LABEL=(2,SL),UNIT=3490
//DD3 DD DSNAME=Y,DISP=NEW,VOL=SER=TAPE01,
//       LABEL=(3,SL),UNIT=3490
```

In JOB2, data set W, after its redirection, is an existing SMS-managed data set. The system does not include data set W in the data set collection. The system does detect data set stacking between DD2 and DD3; data set X and Y make up the data set collection.

If you replace VOL=SER in DD2 with VOL=REF=W and VOL=SER in DD3 with VOL=REF=X, the ACS routines will have the information they need to allocate data sets X and Y to a consistent device category with data set W.

Allocation of Direct Access Space

You must request space for every non-VSAM or non-SMS-managed data set being created on a direct access volume. To tell the system how much space is needed and let the system assign the tracks, code:

```
//ddname  DD  SPACE=(TRK,(primary-qty,second-qty,directory or index)),...
//ddname  DD  SPACE=(CYL,(primary-qty,second-qty,directory or index)),...
//ddname  DD  SPACE=(blklgth,(primary-qty,second-qty,directory or index)),...
```

To tell the system the specific tracks to assign to the data set, code:

```
//ddname  DD  SPACE=(ABSTR,(primary-qty,address,directory or index)),...
```

With SMS, you can request space or override the space allocation defined in the data class for the data set. In this case, code:

```
//ddname  DD  SPACE=(reclgth,(primary-qty,second-qty,directory)),
            AVGREC=value,...
```

Also with SMS, you can code the DATACLAS parameter (or let an ACS routine select a data class) to specify space allocation, for example:

```
   //ddname  DD  DATACLAS=dataclass-name,...
```

Requesting System Assigned Space

Letting the system assign the specific tracks is easiest and most frequently used. Specify only how the space is to be measured — in tracks, cylinders, blocks, or records — and how many of those tracks, cylinders, blocks, or records are required.

Requests for Blocks (blklgth)

Without SMS, it is easiest to specify an average block length: the system allocates the least number of tracks required to contain the number of blocks specified. Specifying block length also maintains device independence; you can change the device type in the UNIT parameter without altering the space request or you can

code in the UNIT parameter a group name that includes different direct access devices.

When you request space in terms of average block length or average record length, the system allocates tracks to contain the request. However, if you code ROUND as the last subparameter in the SPACE parameter, the system allocates the smallest number of cylinders needed to contain the request.

The system allocates DASD space in whole tracks. The number of tracks required depends on how the records are blocked. The system will use one of the following as the block length to compute the number of tracks to allocate, in the order indicated:

1. 4096, if the data set is a PDSE
2. The blocksize from the DCB parameter, if specified
3. The system determined blocksize, if available
4. A default value of 4096.

Requests for Tracks or Cylinders (TRK or CYL)

You can specify TRK or CYL. You will need to compute the number of tracks or cylinders required. Consider such variables as the device type, track capacity, tracks per cylinder, cylinders per volume, data length (blocksize), key length, and device overhead. These variables and examples of estimating space requirements for partitioned and indexed sequential data sets are described in *DFSMS/MVS Using Data Sets*.

Cylinder allocation (and therefore ROUND used with average block or average record) allows faster input/output of sequential data sets than does track allocation.

Requests for Records (reclgth)

With SMS, specify an average record length in bytes, as well as the primary, secondary, and directory quantity on the SPACE parameter, to request space or to override the space allocation in the data class of the data set.

You must also specify the AVGREC parameter with the SPACE parameter in order to specify a record request and indicate whether the primary and secondary quantity represents units, thousands, or millions of records.

How the System Satisfies the Primary Space Request

Space on One Volume

Enough space must be available on one volume to satisfy the primary request. If not, the system terminates the job or searches another volume, depending on the type of volume request made:

Specific volume request: If the first volume specified does not have enough space available, the job is terminated. When extending a multivolume data set, if enough space is not available to satisfy the secondary allocation on the second volume specified, the job is terminated.

Nonspecific volume request: If the first volume chosen by the system does not have enough space available, the system chooses another volume and continues to search for space, asking for volumes to be mounted if necessary. The system continues to search for space until it finds a volume with enough space, exhausts all eligible volumes, or the operator cancels the job.

Note: For a new indexed sequential data set, if the first volume chosen by the system does not contain enough space for the request, the system does not try to find space on another volume, if the request is as follows:

- A request for multiple volumes or units.
- A request is for the second, third, or subsequent DD statement used to define the data set.

Extents

The system tries to allocate the primary and secondary quantity in contiguous tracks or cylinders. If contiguous space is not available, the system satisfies the request with up to five noncontiguous **extents** (blocks) of space.

Multivolume Data Sets

When creating a multivolume data set, the primary quantity cannot specify more space than is available on the first volume. If the primary quantity requests all of the available space on the first volume, the secondary quantity requests space on the subsequent volumes.

Primary Requests in Blocks

If you request space in terms of average block length, the system will compute and allocate the smallest number of tracks (or cylinders if ROUND is specified) to contain the number of blocks specified in primary-qty. blklgth will be used as the block length in this computation, with the exception of the value zero. If a blklgth of zero is specified for the first subparameter, the system uses one of the following as the block length to compute the number of tracks to allocate, in the order indicated:

1. The blocksize from the DCB parameter, if specified
2. The system determined blocksize, if available
3. A default value of 4096.

System Assigned Space Requests with User Labels

If user labels are specified, LABEL=(,SUL), the system allocates up to four noncontiguous extents of space. The system allocates, separately from the primary quantity, one track for user labels. This one track is considered an extent.

How the System Satisfies the Secondary Space Request

For many data sets, the primary quantity does not need to be big enough for the entire data set. Code a **secondary quantity** to be used only if the data set exceeds its originally allocated space. The system tries to obtain the secondary space contiguous to the last extent of space allocated to the data set. But, if you specify the secondary quantity in cylinders, in blocks, or in records with the ROUND subparameter, then the secondary space allocated to the data set starts at the beginning of a cylinder.

Note: BDAM data sets cannot be extended.

Volume for Secondary Space for NEW or MOD Data Set

For data sets whose disposition is NEW or MOD, the system allocates this space on the same volume as the primary quantity until one of the following occurs:

- The volume does not have enough space available for the secondary quantity.
- 16 extents, less the number of extents for the primary quantity and user label space, have been allocated to the partitioned data set (PDS).
- 123 extents, less the number of extents for the primary quantity and user label space, have been allocated to the partitioned data set extended (PDSE).

Then, the system allocates the secondary quantity on another volume only if the DD statement requested more than one volume in the VOLUME parameter or, for a specific volume request, requested more volumes than devices (which implies unit affinity).

If the DD statement makes a nonspecific volume request and the system could possibly allocate a permanently resident volume, code PRIVATE in the VOLUME parameter.

Volume for Secondary Space for OLD Data Set

When allocating a secondary quantity for a data set whose status is OLD, that is, an existing data set being written over or a preallocated data set, the system checks for a next volume. If a next volume exists, the system looks for a secondary quantity already allocated in it. If the system finds a secondary quantity, the system uses that space. If the system finds no space already allocated, the system allocates the secondary quantity on that next volume. If a next volume does not exist, the system allocates the secondary space on the current volume.

Secondary Request Only for Current Execution

A secondary quantity can be requested when creating a data set or when retrieving an existing data set, whether or not you coded a secondary quantity in the original request. A secondary request for an existing data set is in effect only for the duration of the job step and overrides an original request, if one was made.

Secondary Requests in Blocks

If you request space in terms of average block length, the system allocates the least number of tracks required to contain the request.

Directory Space for Partitioned Data Sets

To create a partitioned data set (PDS), request a primary quantity large enough to include space for a directory. A directory is an index used by the system to locate members in the partitioned data set. It consists of 256-byte records, each containing directory entries. There is one directory entry for each member. The third quantity in the SPACE parameter must specify how many records the directory is to contain.

The directory is written at the beginning of the primary space. It must fit in the first extent of the data set. Request enough directory space to allow for growth of the data set. You cannot lengthen the directory once the data set is created. If the directory runs out of space, you must recreate the data set.

With SMS, you can specify the number of records for the directory on the SPACE parameter without specifying any other subparameters. For example:

```
//DD12  DD   DSNAME=PDS.EXMP,DATACLAS=DCLAS12,SPACE=(,(,,20)),
//         DISP=(NEW,KEEP)
```

For a complete description of the directory, including details on member entries to enable you to compute how many records to request, see *DFSMS/MVS Using Data Sets*.

Directory Space for Partitioned Data Sets Extended

The size of a PDSE grows dynamically. If you specify directory size on the SPACE parameter, SMS uses the size you specify only if you later convert the PDSE to a PDS.

System Assigned Space Requests for Indexed Sequential Data Sets

For an indexed sequential (ISAM) data set, space must be requested in cylinders. (Note that SMS does not manage ISAM data sets.)

If you are creating an indexed sequential data set that occupies more than one cylinder and you are not defining the index on a separate DD statement, request index space as the third quantity in the SPACE parameter. The system determines if the third quantity is for a directory or an index from the DCB parameter on the DD statement: DCB=DSORG=IS or DCB=DSORG=ISU must be specified when defining an indexed sequential data set. The system adds the index quantity to the primary quantity when allocating space.

Example 1

```
//ALLO     JOB  (3416,354),STONER,MSGLEVEL=1,MSGCLASS=C
//STEP1    EXEC PGM=TESTSYS0
//DD1      DD   UNIT=3350,DISP=(,PASS),SPACE=(TRK,(10,5))
//DD2      DD   UNIT=3330,DISP=(,PASS),SPACE=(TRK,(10,5))
//SYSABEND DD   SYSOUT=L
//STEP2    EXEC PGM=TESTSYS0
//DD3      DD   DSNAME=*.STEP1.DD1,DISP=(OLD,DELETE,DELETE)
//DD4      DD   VOLUME=REF=*.STEP1.DD2,SPACE=(TRK,(3,1)),UNIT=3330
//SYSABEND DD   SYSOUT=L
```

The first step requests space for two temporary data sets. The second step refers to these data sets for volume information. The space requested for DD1 and DD2 in STEP1 is 10 primary and 5 secondary tracks and for DD4 in STEP2 3 primary and 1 secondary tracks.

Example 2

```
//SMSALLO  JOB   (3444,355),SCHOER,MSGLEVEL=1,MSGCLASS=C
//STEP5    EXEC  PGM=TESTSYS0
//DD6      DD    DSNAME=HIJK.PGM,DATACLAS=DCLAS1,SPACE=(128,(5,2)),
//          AVGREC=K,DISP=(NEW,KEEP)
```

The space requested in DD6 for the new data set overrides the space allocation in the data class for the data set. The space requested is an average record length of 128 bytes, a primary quantity of 5K (5,120) records, and a secondary quantity of 2K (2,048) records.

Requesting Specific Tracks

Requesting that the system allocate specified tracks to a data set (by using the ABSTR subparameter) is the most stringent request for space. If any of the requested tracks on the volume are occupied, the space cannot be allocated and the job is terminated.

Do not code ABSTR for SMS-managed data sets.

Certain uses of certain devices can require that specific tracks be requested. For example, specific tracks must be allocated to position a data set under the fixed heads of a 3348 Model 70F Data Module (cylinders 1-5).

Specific Track Requests with User Labels

If user labels are specified, LABEL=(,SUL), the user labels are placed on a user label track. This track is the first in the space requested.

Specific Track Requests for Indexed Sequential Data Sets

If defining an indexed sequential data set, the number of tracks for the index, primary, or overflow areas must be equal to an integral number of cylinders and on a cylinder boundary. All of the DD statements defining the indexed sequential data sets must request specific tracks.

Example

```
//DDEX  DD  SPACE=(ABSTR,(5,1)),...
```

This example allocates 5 tracks for a data set: beginning at the second track on a volume.

Allocation of Virtual I/O

Temporary data sets can be handled by a facility called virtual input/output (VIO). VIO data sets reside in the paging space; but, to the problem program and the access method, the data sets appear to reside on a direct access storage device.

You cannot use VIO for permanent data sets, indexed sequential data sets, VSAM data sets, or partitioned data sets extended (PDSEs).

VIO provides two advantages:

- VIO speeds reading or writing of a data set. All reading and writing operations are done at the speed of main storage access rather than at the speed of I/O to a device.
- The virtual data set does not occupy space in the user's private area. Thus, unlike a large data area in a program, a virtual data set does not use up program space.

Only the job that creates a VIO data set has access to it to read and write data and to scratch the data set.

SMS manages a VIO data set if (1) you specify a storage class for the data set with the STORCLAS parameter or (2) an installation-written automatic class selection (ACS) routine selects a storage class for the data set. An SMS-managed

VIO data set requires that the assigned storage class supports VIO data sets. Check with your storage administrator.

Requesting VIO

To request a VIO data set, code a DD statement as follows:

- The DSNAME parameter can be coded or omitted. If coded, it must specify a temporary data set:

  ```
  DSNAME=&&dsname
  DSNAME=&&dsname(member)
  ```

- The DISP parameter can be coded or omitted. If coded, it must specify:

  ```
  DISP=(NEW,DELETE)
  DISP=(NEW,PASS)
  DISP=(,PASS)
  ```

- Code a UNIT parameter for non-SMS-managed data sets. UNIT must specify a VIO unit name. During system initialization the installation must define new and/or existing unit names as VIO; the installation should maintain a list of the VIO unit names.

  The unit count subparameter is ignored, if coded.

- The VOLUME parameter can be coded or omitted. If coded, do not specify volume serial numbers.

- The SPACE parameter can be coded or omitted. If coded, the parameter can request up to the size of the simulated volume. The system will allocate as the primary quantity plus 15 secondary quantities an entire simulated volume.

  If the requested primary quantity is larger than the simulated volume, the job will fail. If the primary request is met, but the secondary request is greater than one volume, the system allocates up to one volume. When allocating by average block length for a VIO data set, the secondary request is computed using the average block length specified in the SPACE parameter.

  If the SPACE parameter is omitted, the system uses a default value: 4 primary and 24 secondary blocks, with an average block length of 8192. If the VIO data set is directed to SMS and space values are specified for the data class chosen by the ACS routines, the data class values will take effect rather than the allocation defaults.

- The DCB parameter can be coded or omitted. If coded, do not specify IS or ISU in the DSORG subparameter.

The system will allocate a VIO data set request to actual direct access storage if the DD statement contains unacceptable parameters; however, if the primary quantity is too big, the system terminates the job.

Example 1

```
//EX1  DD  UNIT=VIO
//EX2  DD  DSNAME=&&TEMPDS,UNIT=SYSDA
//EX3  DD  DSNAME=&&TEMPDS(MEM1),UNIT=VIRT3
//EX4  DD  DSNAME=&&MYDS,UNIT=VIO,SPACE=(360,(5,30)),
//      DISP=(,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=360)
```

In these examples, the system assigned during system initialization the group names VIO, SYSDA, and VIRT3 as eligible for VIO processing.

Example 2

```
//EXSMS DD  DSNAME=&&TEMP,STORCLAS=SCLASVIO
```

In the example, EXSMS defines an SMS-managed VIO data set because the storage administrator has defined storage class SCLASVIO with support for VIO.

Backward References to VIO Data Sets

If a DD statement defines a temporary data set and refers in a VOLUME=REF parameter to a DD statement for a VIO data set, the system assigns the data set to external page storage as a VIO data set.

If a DD statement requests unit affinity to a VIO data set but does not define a temporary data set, the system allocates the data set to the VIO unit but does not assign it VIO status.

The examples assume that the installation defined during system initialization the group name SYSDA and the device type name 3330 as eligible for VIO processing. Except where noted, all of the following DD statements cause allocation of VIO data sets.

Example 1

```
//DD1  DD  UNIT=SYSDA
```

Example 2

```
//DD2  DD  UNIT=3330
```

Example 3

```
//DD3  DD  DSNAME=&&A,DISP=(NEW),SPACE=(CYL,(30,10)),UNIT=SYSDA
```

Example 4

```
//DD1  DD  UNIT=SYSDA
//DD2  DD  VOLUME=REF=*.DD1
```

Example 5

```
//DDA  DD  UNIT=SYSDA
//DDB  DD  VOLUME=REF=*.DDA,UNIT=3330
```

Example 6

```
//DD1  DD  UNIT=SYSDA
//DD2  DD  DSNAME=NONTEMP,DISP=(,KEEP),
//       VOLUME=REF=*.DD1,SPACE=(CYL,10)
```

In this example, the data set defined in DD1 is assigned to external page storage for VIO processing. Because DD2 defines a permanent data set, the system assigns it to direct access storage.

Example 7

```
//DD1  DD  UNIT=SYSDA
//DD2  DD  DSNAME=&&TEMP,VOLUME=SER=665431,
//       SPACE=(CYL,10),UNIT=AFF=DD1
```

In this example, the data set defined in DD1 is assigned to external page storage for VIO processing. Because DD2 specifies a volume serial number, the system assigns it to direct access storage.

Example 8

```
//REGJOB    JOB  3344,'DEPT. 28'
//ASM       EXEC PGM=IFOX00
              .
              .
//ASM.SYSGO DD   DSNAME=&&OBJ,UNIT=VIO,DISP=(NEW,PASS)
//LKED      EXEC PGM=IEWL
//SYSLIN    DD   DSNAME=&&OBJ,DISP=(OLD,DELETE)
//           DD   DDNAME=SYSIN
//SYSLMOD   DD   DSNAME=&&LOAD(A),DISP=(NEW,PASS),UNIT=VIO,
//            DCB=DSORG=PO,SPACE=(TRK,(5,5,1))
              .
              .
//GO        EXEC PGM=*.LKED.SYSLMOD
```

VIO data sets are passed in the same way as conventional data sets. This example shows the DD statements for VIO data sets in a job whose steps compile and link edit a program and then execute that program. The three VIO data sets are defined in the statements ASM.SYSGO, SYSLIN, and SYSLMOD.

Note: The SPACE parameter must appear on the //SYSLMOD DD statement to make sure that directory space is allocated.

Allocation with Volume Premounting in a JES2 System

In a JES2 system, to identify volumes that the operator must mount before the job is executed, code:

```
/*SETUP serial-number,...
```

When the job enters the system, JES2 issues a message to the operator console to ask the operator to mount the identified volumes. JES2 places the job on hold until the operator mounts the volumes, then releases the job.

Example

```
/*SETUP  223344,556677,889900
```

Note: IBM recommends that you do not use the /*SETUP control statement to specify volumes in an IBM 3495 Tape Library Dataserver. This statement causes the job to be unnecessarily held until released by the operator.

Dynamic Allocation

Dynamic allocation allows a job to acquire resources as they are needed and release them immediately after use. The resources are a ddname-data set combination with its volumes and devices.

One reason to use dynamic allocation is that you may not know all of the device requirements for a job before execution. Another reason is that it allows the system to use resources more efficiently; that is, the system can acquire resources just before their use and release them immediately after use.

To tell the system the number of resources to be held in anticipation of reuse, code:

```
//stepname EXEC PGM=x,DYNAMNBR=n
```

The system uses the sum of this number and the number of DD statements in the step to establish a control limit for tracking resources that it is holding in anticipation of reuse.

For more information on dynamic allocation, see *OS/390 MVS Programming: Authorized Assembler Services Guide*.

Example

```
//PROS   JOB  1585,SALLYJ,CLASS=A,PERFORM=70
//STEP1  EXEC PGM=TEST,DYNAMNBR=4,PARM=(P3,123,MT5)
//OUT1   DD   SYSOUT=C,FREE=CLOSE
//OUT2   DD   SYSOUT=A
//SYSIN  DD   *
  .
  .
  data
  .
/*
```

- The JOB statement specifies that this job will be processed in class A and in performance group 70.
- The control limit is **the sum of the number of DD statements coded** and **the value coded in the DYNAMNBR parameter**:

  ```
  3 DD statements + 4 = 7
  ```

  If this control limit is reached and another dynamic allocation is requested, the request is not honored unless resources can be unallocated so that the control limit is not exceeded.
- When OUT1 is closed, it is immediately ready for printing.

Chapter 16. Data Set Resources - Processing Control

Figure 16-1. Processing Control Task for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Processing Control					
by suppressing processing	DUMMY NULLFILE on DSNAME				
by postponing specification	DDNAME				
with checkpointing	CHKPT SYSCKEOV DD				
by subsystem	SUBSYS CNTL		CNTL ENDCNTL		
by TCAM job or task	QNAME				

Processing Control by Suppressing Processing

To suppress processing of a data set, assign it a dummy status by coding either of the following:

```
//ddname  DD  DUMMY,...
//ddname  DD  DSNAME=NULLFILE,...
```

The system ignores all parameters other than DUMMY or DSNAME=NULLFILE and DCB. The DCB parameter must be coded if you would code it for normal I/O operations. For example, when an OPEN routine requires a BLKSIZE specification to obtain buffers and BLKSIZE is not specified in the DCB macro instruction, code this information in the DD DCB parameter.

Effect of Dummy Data Set

For a dummy data set, the system bypasses all input/output operations, does not allocate devices or storage to the data set, and does not perform disposition processing.

Requests to Read or Write a Dummy Data Set

When the program asks to read a dummy data set, an end-of-data-set exit is taken immediately. When the program writes to the dummy data set, the request is recognized but no data is transmitted. VSAM supports dummy data sets for both read and write processing. BSAM and QSAM support requests to write to a dummy data set. If any other access method is used, the job is terminated.

Use of Dummy Data Sets

© Copyright IBM Corp. 1988, 1999

When testing a program, you can suppress writing of an output data set by defining it as a dummy data set. This would forestall printing a data set until you are sure it contains meaningful output.

To save processing time, you might not want a data set to be processed every time the job is executed. For example, you might want to skip reading a data set that is used only once a week.

Nullifying a Dummy Data Set

When the data set is to be processed, replace the DD statement that specified the dummy data set with a DD statement containing the parameters required to define the data set. When a procedure DD statement specifies a dummy data set, nullify it by coding the DSNAME parameter on the overriding DD statement and assigning a data set name other than NULLFILE.

Examples

```
//EXA  DD  DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//      UNIT=3211
//EXB  DD  DSNAME=NULLFILE,UNIT=DISK,VOLUME=SER=165789,
//      DISP=OLD
//EXC  DD  DUMMY,DISP=OLD
```

Processing Control by Postponing Specification

To postpone specification of a data set, reference a later DD statement by coding:

```
//ddname  DD  DDNAME=ddname
```

How the System Postpones Data Set Definition

When the system encounters a DD statement with a DDNAME parameter, it saves the ddname and, temporarily, the name in the DDNAME parameter; the system uses the DDNAME name to relate the statement to a later DD statement. When the system finds a statement whose ddname has been temporarily saved, it does the following:

- It uses the parameters on the statement with the matching ddname to define the data set.
- It associates these parameters with the name of the statement that contained the DDNAME parameter.
- It stops saving the name from the DDNAME parameter.

References to the Data Set

The system associates the ddname of the statement that contains the DDNAME parameter with the data set definition. The system does not use the ddname of the later statement that actually defines the data set. Therefore, any references to the data set, before or after the data set is defined, must refer to the DD statement that contains the DDNAME parameter, not the referenced DD statement that defines the data set.

Concatenating DD Statements when DDNAME is Specified

To concatenate data sets to a data set defined with a DDNAME parameter, the unnamed DD statements must follow the DD statement that contains the DDNAME parameter, not the referenced DD statement that defines the data set.

Use of Postponing Specification

Use the DDNAME parameter in cataloged procedures to postpone defining an in-stream data set until a job step calls the procedure. Procedures cannot contain DD statements that define in-stream data sets and cannot contain in-stream data.

Use the DDNAME parameter in a job step that calls a procedure to postpone defining in-stream data until the last overriding DD statement for a procedure step. Overriding DD statements must appear in the same order as the DD statements in the procedure and any in-stream data sets must appear last in a calling step.

Example 1

```
//XYZ   DD  DDNAME=PHOB
        .
        .
        .
//PHOB  DD  DSNAME=NIN,DISP=(NEW,KEEP),UNIT=3400-5
```

From DD statement XYZ, the system saves XYZ and, temporarily, PHOB. Until the system encounters the ddname PHOB, it treats the data set for XYZ as a dummy data set.

When the system reads DD statement PHOB, it uses the DSNAME, DISP, and UNIT values to define the data set named NIN. The system also associates this information with DD statement XYZ. The system stops saving ddname PHOB. The data set is now defined as if you had coded:

```
//XYZ  DD  DSNAME=NIN,DISP=(NEW,KEEP),UNIT=3400-5
```

Example 2

```
//DD1  DD  DDNAME=LATER
   .
   .
   .
//LATER  DD  DSN=SET12,DISP=(NEW,KEEP),UNIT=3350,
//           VOLUME=SER=46231,SPACE=(TRK,(20,5))
   .
   .
   .
//DD12  DD  DSN=SET13,DISP=(NEW,KEEP),VOLUME=REF=*.DD1,
//          SPACE=(TRK,(40,5))
```

DD1 postpones defining the data set until the system encounters DD statement LATER. DD12 must do a backward reference to DD1 because the system associates the data set information with the DD statement that contains the DDNAME parameter.

Example 3

```
//DDA  DD  DDNAME=DEF
//     DD  DSN=A.B.C,DISP=OLD
//     DD  DSN=SEVC,DISP=OLD,UNIT=3350,VOL=SER=52226
   .
   .
   .
//DEF  DD  *
   data
/*
```

This example shows correct concatenation when a DDNAME parameter is coded.

Processing Control with Checkpointing

To write a checkpoint when the system reaches an end of volume while processing a multivolume input or output data set, code:

```
//ddname  DD  CHKPT=EOV,...
```

The system writes checkpoints for all volumes but the last. The data set must be a multivolume QSAM or BSAM data set. Checkpoints are not written for single-volume QSAM or BSAM data sets or for ISAM, BDAM, BPAM, or VSAM data sets.

The system writes the checkpoints in a SYSCKEOV data set. A SYSCKEOV DD statement must be specified in a step with a DD statement that contains CHKPT and again when the step is restarted from a checkpoint written in the data set.

Examples

```
//S1       EXEC PGM=A,RD=R
//D1       DD   DSNAME=OUT1,UNIT=(DISK,3),DISP=(NEW,CATLG),
//          SPACE=(400,(50,10),VOLUME=(PRIVATE,,,3),CHKPT=EOV
//SYSCKEOV DD   DSNAME=CK1,UNIT=3350,DISP=(MOD,KEEP),
//          SPACE=(CYL,30,,CONTIG)
```

Processing Control by Subsystem

Requesting Subsystem

To ask a subsystem to process a data set and to specify parameters for the subsystem, code:

```
//ddname  DD  SUBSYS=subsystem-name,...
//ddname  DD  SUBSYS=(subsystem-name,subparameter,...),...
```

The subsystem processes the subparameters according to its own rules.

When you specify the SUBSYS parameter, the subsystem may alter the significance of certain DD statement parameters. For details, see the documentation for the subsystem.

If you specify the DUMMY parameter, MVS invokes the subsystem to check the syntax of subsystem subparameters. If the syntax is acceptable, MVS assigns a dummy status to the data set and processes the request as a dummy request.

If you request unit affinity to a subsystem data set, MVS substitutes SYSALLDA as the UNIT parameter specification.

Example

```
//EXSUB  DD  DSNAME=MYSET,DISP=OLD,SUBSYS=(PRO3,34,92)
```

Program Control Statements for a Subsystem

To specify control information for a subsystem, code:

```
//stepname EXEC PGM=x
//label    CNTL
            .
            .
           (program control statements)
            .
//         ENDCNTL
//ddname   DD      SUBSYS=subsystem-name,CNTL=*.label
```

Program control statements supply control information for the subsystem.

Example

```
//S1    EXEC  PGM=REPT
//ABC   CNTL
//PGC   PRINTDEV BUFNO=2-,PIMSG=YES
//      ENDCNTL
//DD1   DD    SUBSYS=XYZ,CNTL=*.ABC
```

Processing Control by TCAM Job or Task

To define a data set of telecommunications access method (TCAM) messages and to ask a TCAM job or started task to process the data set, code:

```
//ddname  DD  QNAME=procname,...
//ddname  DD  QNAME=procname.tcamname,...
```

For more information, see *ACF/TCAM Installation Reference*

Examples

```
//DSA   DD  QNAME=MES34.TJOB,DCB=(RECFM=FB,LRECL=80,BLKSIZE=320)
//DSB   DD  QNAME=MES78.TJOB
```

Chapter 17. Data Set Resources - End Processing

Figure 17-1. End Processing Task for Requesting Data Set Resources

TASKS FOR REQUESTING DATA SET RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
End Processing					
unallocation	FREE				
disposition of data set	DISP RETPD EXPDT				
release of unused direct access space	RLSE on SPACE				
disposition of volume	RETAIN and PRIVATE on VOLUME				

Data sets on system-managed tape library volumes exhibit both system-managed and non-system-managed characteristics. When necessary, **datasets on a system-managed tape volume** are distinguished from **system-managed DASD data sets**. Otherwise, the term **system-managed data sets** refers to both datasets on a system-managed tape volume and system-managed DASD data sets.

Unallocation End Processing

The system unallocates data sets and their associated volume and devices at the end of a job step or at the end of the job.

Dynamic Unallocation

To unallocate a data set while a step is still executing, code:

```
//ddname  DD  FREE=CLOSE,...
```

Use FREE=CLOSE to allow the system to reallocate a volume or device that is used frequently in the system.

Example

```
//DD1 DD DSNAME=DS6,DISP=OLD,UNIT=TAPE,VOLUME=SER=111111,FREE=CLOSE
```

Disposition End Processing of Data Set

Disposition end processing of a data set is controlled by either the DISP parameter or by time, expressed as a retention period (RETPD) or an expiration date (EXPDT).

© Copyright IBM Corp. 1988, 1999

Disposition Controlled by DISP Parameter

The system processes a data set after its use depending on how the step terminates:

- **Normal termination disposition:** To delete, keep, pass, catalog, or uncatalog the data set when the step terminates normally, code:

```
//ddname  DD  DISP=(,DELETE),...
//ddname  DD  DISP=(,KEEP),...
//ddname  DD  DISP=(,CATLG),...
//ddname  DD  DISP=(,UNCATLG),...
//ddname  DD  DISP=(,PASS),...
```

- **Abnormal termination or conditional disposition:** To delete, keep, catalog, or uncatalog the data set if the step terminates abnormally, code:

```
//ddname  DD  DISP=(,,DELETE),...
//ddname  DD  DISP=(,,KEEP),...
//ddname  DD  DISP=(,,CATLG),...
//ddname  DD  DISP=(,,UNCATLG),...
```

You should consider coding an abnormal termination disposition every time you create or use a data set. This disposition can be used to keep data sets after a program fails, when they might be needed to determine the cause of the failure. This disposition can also be used to delete data sets in case of program failure, thereby restoring the system environment to what it was before the error. Then the failing job can be rerun without an intervening clean-up job.

Effect of Abnormal Termination During Execution

When a step abnormally terminates but is not automatically restarted, its data sets are disposed of as specified by the abnormal termination disposition. If an abnormal termination disposition is not specified, the normal termination disposition is processed.

Effect of Abnormal Termination During Allocation

If a job step fails during step allocation, the system disposes of the data sets as follows:

- Deletes a data set being created in the step.
- Keeps a data set that existed before the step.

Effect When No Abnormal Termination Disposition is Coded

If a DD statement in an abnormally terminating step requests a data set that was cataloged or kept in an earlier step and if the statement does not specify an abnormal termination disposition, the system uses the disposition specified in the earlier step.

Effect of Device Type on Disposition

The system handles disposition differently for data sets on direct access and on tape. A direct access volume contains a volume table of contents (VTOC). A VTOC describes the non-VSAM data sets and available space on the volume.

Deleting a Data Set

Specifying DELETE requests that the data set's space on the volume be released at termination of the step:

- If the data set is on a public tape volume, the tape is rewound. The volume is available for use by other job steps.
- If the data set is on a private tape volume, the tape is rewound and unloaded. The system issues a KEEP message.
- If the data set is on a private direct access volume, the description of the data set is removed from the VTOC. The space on the volume is available to other data sets.

Note: If you delete the only remaining data set on a system-managed tape volume, the system does mark the volume as scratch at job or step termination. The storage administrator controls the return of a volume to scratch status.

Unexpired Expiration Date: In one case, however, a data set on a direct access volume is not deleted: If a data set previously existed and has an unexpired expiration date, an abnormal termination disposition of DELETE does not delete the data set if the step abnormally terminates.

Cataloged Data Sets: If you are deleting a cataloged non-VSAM data set, the entry for the data set in the system catalog is removed when the system obtains the volume serial number from the catalog. When the volume serial number is coded or referenced on the DD statement, the data is deleted but its entry remains in the catalog.

If an error occurs while the system is deleting a cataloged data set, its entry remains in the catalog. The data set itself is or is not deleted, depending on when the error occurs.

To delete an entry from a VSAM catalog, use the DELETE command as described in *DFSMS/MVS Using Data Sets*. Using the DELETE command makes the space occupied by the data set available for reallocation. To delete catalog entries for data sets that are not cataloged in a VSAM catalog, use the UNCATLG statement of IEHPROGM as described in *DFSMS/MVS Utilities*.

Temporary Data Sets: DELETE is the only valid abnormal termination disposition for a temporary data set. If you specify a disposition other than DELETE, the system assumes DELETE.

TSO/E Background Data Sets: In a step running TSO/E, the system replaces a DD statement disposition of DELETE with a disposition of KEEP. This prevents an attempt to delete a data set that has been unallocated by the TSO/E FREE command.

Keeping a Data Set

Specifying KEEP instructs the system to keep a data set intact until a later step or job requests that the data set be deleted or cataloged or until after an expiration date or retention period, if specified.

For data sets on direct access, the entry in the VTOC describing the data set and the data set itself are kept. For data sets on tape, the volume is rewound and unloaded, and a KEEP message is issued to the operator.

Note: If you specify KEEP for a temporary data set, the system changes the disposition to PASS. See “Passing a Data Set” on page 17-5 for more information about how the system handles passed data sets.

Cataloging a Data Set

Catalog a non-VSAM or non-system-managed data set, or datasets on a system-managed tape volume, by specifying CATLG as the disposition. (The system automatically catalogs VSAM and system-managed DASD data sets when they are allocated.)

For a new data set, the system keeps the data set and creates an entry pointing to it in one of the following:

- The system-determined catalog, if the step or job does not specify a private catalog.
- The private catalog specified in a STEPCAT DD statement in the step.

  With SMS, do not use the STEPCAT DD statement in a job step that references an SMS-managed data set.
- The private catalog specified in a JOBCAT DD statement in the job, if the step does not contain a STEPCAT DD statement.

  With SMS, do not use the JOBCAT DD statement in a job that references an SMS-managed data set.

For an old data set, the system keeps the data set, and does the following depending on the parameters on the DD statement.

- If UNIT and VOLUME=SER are not coded, the system updates the catalog that is used to locate the data set.
- If UNIT and VOLUME=SER are coded, the system creates a new catalog entry in the applicable system master or private catalog, even if the data set is already cataloged.

A private catalog can be either a VSAM user catalog or an integrated catalog facility catalog.

Use of Cataloging: Cataloging allows you to keep track of the location of data sets. Cataloging also simplifies retrieving a data set: code only the DSNAME parameter and OLD, SHR, or MOD in the DISP parameter and omit volume and device information.

CATLG for a Cataloged Data Set: Specify a disposition of CATLG for an already cataloged data set when adding to the data set if it may need another volume. The system updates the catalog entry to include the volume serial numbers of any additional volumes if the data set was specified as follows:

- DISP=(MOD,CATLG)
- No volume serial numbers were coded or referenced on the DD statement

Generation Data Sets: A collection of cataloged data sets that are kept in chronological order is a generation data group (GDG). The entire GDG is stored under a single data set name; each data set within the group, called a generation data set, is associated with a generation number that indicates how far removed the data set is from the original generation. When creating a new generation data set, code a disposition of CATLG.

When the System Does Not Catalog a Data Set: The system does not catalog a data set if the data set is not opened by the problem program and one of the following is true:

- The DD statement made a nonspecific request for a tape volume.
- The DD statement requested a tape volume for a tape device with dual density options but did not specify the density in the DEN subparameter of the DCB parameter.

Job Termination Due to Inability to Catalog a Data Set: The system terminates the job when the installation option specifies that the job is to be terminated if, during data set disposition processing of a batch unallocated data set, the system is unable to:

- Catalog a new data set for which a disposition of CATLG was specified
- Catalog an old uncataloged data set for which a disposition of CATLG was specified
- Recatalog an old cataloged data set for which the volume list was extended and a disposition of CATLG, KEEP, or PASS was specified
- Roll an SMS-managed generation data set into the GDG base.

The installation options do not apply if, by specifying FREE=CLOSE, the data set is unallocated when closed.

Uncataloging a Data Set

To remove from the catalog the entry describing a non-VSAM or non-system-managed data set, or a data set on a system-managed volume, code UNCATLG as the disposition. Specifying UNCATLG does not delete the data set; only the reference in the catalog is removed. If you request the data set in a later job or step, the DD statement must specify volume information. Note that you cannot use JCL to uncatalog system-managed DASD data sets.

Passing a Data Set

If more than one step in a job needs the same data set, each DD statement for the data set can pass it to a later step. A data set can be passed only within a job. A data set cannot be passed and received within the same step.

To Pass: To pass a data set, code PASS as the normal termination disposition; PASS cannot be the abnormal termination disposition. Code PASS each time the data set is needed until the last use in the job. In the last DD statement for the data set, assign it a final disposition.

To Receive: To receive a passed data set, specify in the DD statement the data set name without specifying a volume serial number or volume reference.

Data sets with identical names, whether or not the names refer to the same data

set, can be passed within the same job. If you receive a data set with a disposition of DELETE, then data sets with identical names are received in the same order in which they are passed. If you receive a data set with a disposition of PASS, then subsequent steps that attempt to receive a data set with that name will experience unpredictable results, including possible loss of data.

Do not try to receive a passed data set more times than it is passed.

When a passed data set is received, the passed data set information is no longer available to later DD statements in the receiving step or later steps. Therefore, a VOLUME=REF parameter that refers to the passed data set must appear on a DD statement before the receiving DD statement. For example:

```
//JEX  JOB   ACCT27,'GALE RUCINSKI'
//S1   EXEC  PGM=A
//DA   DD    DSNAME=MYDATA,DISP=(NEW,PASS),
//           SPACE=(800,15),UNIT=DISK
//S2   EXEC  PGM=B
//DB   DD    DSNAME=REPT,DISP=(NEW,PASS),
//           SPACE=(800,15),UNIT=DISK,VOLUME=REF=MYDATA
//DC   DD    DSNAME=*.S1.DA,DISP=SHR
/*
```

For SMS permanent data sets, the restrictions on receiving passed data sets do not apply. All SMS-managed permanent data sets are cataloged, and can be located using the normal catalog search.

In a JES3 system, if the data set was extended to additional volumes, code UNIT=AFF=ddname in the DD statement that receives the data set. This makes JES3 aware of the additional device needed for the extended data set.

When Passing Step Abnormally Terminates: If a step that passes a data set abnormally terminates during execution, the passed data set is passed. Thus, a following step that is executed because of a COND=EVEN or COND=ONLY can receive and process the passed data set. If the passed data set remains unreceived at the end of the job, the system performs the abnormal termination disposition, if specified, for the passed data set.

Disposition Processing of Unreceived Passed Data Sets: If a passed data set is never received by a later step, at the end of the job the system processes the data set as an unreceived, passed data set. This can result in unintentionally deleting the data set, even if it had been cataloged during the job, as the following example shows.

EXAMPLE:

Data set “dsname,” which does not exist at the start of a job but is created and cataloged during the job, will be uncataloged and deleted if it is passed and not received:

```
//Step1  EXEC  PGM=pgmname1
//DD1    DD    DSN=dsname,DISP=(NEW,CATLG,DELETE)
//*
//Step2  EXEC  PGM=pgmname2
//DD2    DD    DSN=dsname,DISP=(OLD,PASS,DELETE)
//*
//Step3  EXEC  PGM=pgmname3
//Step4  EXEC  PGM=pgmname4
//DD4    DD    DSN=dsname,DISP=(OLD,PASS,DELETE)
```

Data set “dsname” is cataloged when Step1 ends. After Step2 ends, “dsname” is still cataloged. If Step3 terminates abnormally, “dsname” will be deleted during end of job processing, because it had been passed by Step2 and not received by a following step, AND the abnormal disposition for Step2 was DELETE.

To avoid that situation, do not specify PASS for a cataloged data set—no matter whether it had been created in a prior job or in a prior step of this job. The correct JCL is:

```
//Step1  EXEC  PGM=pgmname1
//DD1    DD    DSN=dsname,DISP=(NEW,CATLG,DELETE)
//*
//Step2  EXEC  PGM=pgmname2
//DD2    DD    DSN=dsname,DISP=(OLD,KEEP,DELETE)
//*
//Step3  EXEC  PGM=pgmname3
//Step4  EXEC  PGM=pgmname4
//DD4    DD    DSN=dsname,DISP=(OLD,KEEP,DELETE)
```

At Abnormal Termination when Abnormal Termination Disposition is Specified: If a job step abnormally terminates, unreceived data sets that specified an abnormal termination disposition when passed are processed according to the specifications in their abnormal termination dispositions.

For example, you code DISP=(,PASS,CATLG) for a new data set. If this step, or a later step before the receiving step, abnormally terminates during execution, the system tries to catalog the data set as instructed by the abnormal termination disposition of CATLG.

Non-system-managed data sets and datasets on a system-managed tape volume are not processed as specified in their abnormal termination dispositions. If the abnormal termination disposition requires an update to a private catalog and:

1. CATLG is specified for a data set that has a first-level qualifier of a catalog name or alias, the system does not catalog the data set.
2. UNCATLG or DELETE of a cataloged data set is specified for a data set that has a first-level qualifier of a catalog name or alias, the system does not uncatalog the data set.

3. CATLG is specified for a data set that does not have a qualifier or has a qualifier that is not a catalog name, the system catalogs the data set in the master catalog.
4. UNCATLG or DELETE of a cataloged data set is specified for a data set that does not have a qualifier or has a qualifier that is not a catalog name, the system tries to uncatalog the data set from the master catalog.

At Abnormal Termination when No Abnormal Termination Disposition is Specified: If no job step abnormally terminates before it begins execution, the system deletes all unreceived passed data sets that specified (NEW,PASS) and that did not specify an abnormal termination disposition; the system keeps all others. The system deletes those data sets even if they have unexpired expiration dates or retention periods.

When Abnormal Termination Occurs Before Execution: If a step abnormally terminates before it actually begins execution, for example, during allocation of devices and volumes or direct access space, the system ignores the disposition on the DD statement. The system keeps existing data sets and deletes new data sets.

Deletion at End of Job: If unreceived passed data sets are deleted at the end of a job, the system performs dynamic allocation to allocate a device and volume for deletion. Depending on the JOB statement MSGLEVEL parameter or the installation defaults, the system issues allocation messages for these data sets.

In a Procedure That is Called Multiple Times: A problem can occur when the same data set is passed more times than it is received in a procedure that is called more than once in a job. This is illustrated by the following example:

```
Cataloged procedure MYPROC:

  //STEP1  EXEC PGM=IEFBR14
  //DD1    DD   DSNAME=&A,DISP=(NEW,PASS),
  //            SPACE=(TRK,(1,1)),UNIT=SYSDA
  //DD2    DD   DSNAME=*.DD1,DISP=(OLD,PASS),
  //            VOL=REF=*.DD1
  //STEP2  EXEC PGM=IEFBR14
  //DD3    DD   DSNAME=&A,DISP=(OLD,DELETE)

Input stream:

  //JOBEX  JOB
  //S1     EXEC PROC=MYPROC
  //S2     EXEC PROC=MYPROC
```

DD1 and DD2 pass data set &A. DD3 receives data set &A. After the procedure has been executed, one entry for data set &A remains unreceived.

When the procedure is called a second time, DD3 receives data set &A from the first execution of the procedure. This can result in incorrect data or an abnormal termination. If data set &A is not received twice in the job, data set &A is processed as an unreceived passed data set at the end of the job.

Default Disposition Processing

If you omit the DISP parameter or one of its subparameters, the system supplies default values.

If the data set status is omitted, the system assumes NEW. If the second or third subparameter is omitted, the system determines how to handle the data set according to the status of the data set:

- Data sets that existed before the job are automatically kept. The system treats a data set as existing when the status is OLD, SHR, or MOD with volume information.
- Data sets created in the job are automatically deleted. The system treats a data set as newly created when the status is NEW, omitted, or MOD without volume information.

Bypassing Disposition Processing

If you define a data set as a dummy data set, the system ignores the DISP parameter, if coded, and does not perform disposition processing.

Disposition Processing of Data Sets that Do Not Exist

When you code a status subparameter of OLD, SHR, or MOD on a DD statement for a data set that does not exist, processing proceeds based on whether you have supplied VOLUME and UNIT information on the DD statement.

When VOLUME and UNIT Are Coded: When you code VOLUME and UNIT on the DD statement, a JCL error will occur if the problem program attempts to open the data set. Otherwise, the data set disposition depends on the DISP normal termination disposition:

- When the normal termination disposition is **KEEP**, the job log will show that the data set was kept.
- When the normal termination disposition is **CATLG**, and a catalog entry exists for the data set name, you will receive an error message stating that the data set was not recataloged.

  When no catalog entry exists for the data set name, and you have provided the unit information, volume serial, and, for tape data sets, recording mode or density, the system will catalog the data set. For tape data sets, without proper density or recording mode information (when density and recording mode are required), you will receive an error message that the data set was not cataloged.
- When the normal termination disposition is **UNCATLG**, and a catalog entry exists for the data set name, the system will uncatalog the data set.

  When no catalog entry exists for the data set name, you will receive an error message stating that the data set was not uncataloged.
- When the normal termination disposition is **PASS**, the system passes the data set.
- When the normal termination disposition is **DELETE**, the job log will show that the system did not delete the data set. However, this does not affect the job step condition code or produce a JCL error.

When VOLUME and UNIT Are Not Coded: When you do not code VOLUME and UNIT on the DD statement, and the data set is not cataloged, you will receive a JCL error. If the data set is cataloged, and the problem program attempts to open the data set, you will receive a JCL error. If the data set is cataloged and the problem program does not attempt to open the data set, the disposition depends on the DISP normal termination disposition.

- When you code a normal termination disposition of **KEEP, CATLG, UNCATLG**, or **PASS**, the data set disposition is the same for each of these subparameters as described in “When VOLUME and UNIT Are Coded” on page 17-9.
- When you code a normal termination disposition of **DELETE**, the system will uncatalog the data set. The job log will show that the data set was not deleted.

Example 1

```
//DISPJ JOB  158765,'SECT. 27'
//S1    EXEC PGM=IEFBR14
//D1    DD   DSN=ABC,DISP=(SHR,KEEP)
//D2    DD   DSN=SYSA,DISP=(OLD,DELETE,UNCATLG)
//D3    DD   DSN=SYSB,UNIT=3350,VOL=SER=335001,
//           SPACE=(CYL,(4,2,1)),DISP=(NEW,CATLG,KEEP)
//D4    DD   DSN=&&SYS1,DISP=(MOD,PASS),UNIT=3350,
//           VOL=SER=335004,SPACE=(TRK,(15,5,1))
//S2    EXEC PGM=IEFBR14
//D5    DD   DSN=&&SYS1,DISP=(MOD,DELETE),UNIT=3350,
//           VOL=SER=335004,SPACE=(TRK,(15,5,1))
```

1. D1 requests a data set that already exists and can be shared with other jobs. It is to be kept on the volume at the end of step S1.
2. D2 requests a data set that already exists and cannot be shared with other jobs. It is to be deleted at the end of S1, but is to be kept and uncataloged if S1 abnormally terminates.
3. D3 defines a new data set that is to be assigned to volume 335001 on a 3350 Direct Access Storage device. The data set is to be kept on the volume and cataloged if S1 terminates normally, but is to be kept and not cataloged if S1 terminates abnormally.
4. D4 defines a temporary data set that is to be created in this job step. It is to be assigned to volume 335004 on a 3350 and allocated 15 primary tracks, five secondary tracks, and one directory record. This data set is to be passed for use in a later step in this job.
5. D5 requests the temporary data set passed by D4 of S1. When S2 completes, the data set is to be deleted.

Example 2

```
//PASS JOB  ,'BILL H.'
//S1   EXEC PGM=IEFBR14
//DD1  DD   DSN=A,DISP=(NEW,PASS),VOL=SER=335000,
//          UNIT=3350,SPACE=(TRK,1)
//DD2  DD   DSN=A,DISP=(OLD,PASS),VOL=REF=*.DD1
//DD3  DD   DSN=B,DISP=(OLD,PASS),VOL=SER=335000,UNIT=3350
//DD4  DD   DSN=B,DISP=(OLD,PASS),VOL=SER=335001,UNIT=3350
//S2   EXEC PGM=IEFBR14
//DD5  DD   DSN=A,DISP=OLD
//DD6  DD   DSN=A,DISP=OLD
//DD7  DD   DSN=B,DISP=OLD
//DD8  DD   DSN=B,DISP=(OLD,PASS)
//S3   EXEC PGM=IEFBR14
//DD9  DD   DSN=B,DISP=OLD
```

1. DD1 and DD2 pass the same data set. DD5 and DD6 receive that same data set.
2. DD3 and DD4 pass different data sets of the same name. DD7 receives the data set passed by DD3; DD8 receives the data set passed by DD4. DD8 continues to pass the data set originally passed by DD4.
3. DD9 receives the data set passed by DD8.

Disposition Controlled by Time

When you create a data set, tell the system how long to keep it by coding a retention period or an expiration date.

Use the RETPD or EXPDT DD parameter:

```
//ddname  DD  RETPD=nnnn,...
//ddname  DD  EXPDT=yyddd,...
    or
//ddname  DD  EXPDT=yyyy/ddd,...
```

As long as the time period has not expired, the system will not delete or write over a data set on direct access space. This is true even if a DD statement specifies a disposition of DELETE (other than DISP=(NEW,DELETE)) for the data set. The data set is eligible for deletion once the expiration date or retention period has been reached.

When the expiration date of a data set is the current date, the data set is considered expired. The system will delete it or write over it if requested in a DD statement.

Deleting before Expiration Date or Retention Period

If it is necessary to delete a data set before the expiration date or retention period, do one of the following:

- For data sets cataloged in a VSAM catalog, use the DELETE command; this makes the space occupied by the data set available for reallocation. See

  DFSMS/MVS Access Method Services for ICF
  DFSMS/MVS Access Method Services for VSAM

- For data sets cataloged in a non-VSAM catalog, delete the catalog entry with the IEHPROGM utility as described in *DFSMS/MVS Utilities*.

- For the data set control block, use a SCRATCH macro with the OVRD parameter; this makes the space occupied by that data set available for reallocation. See *DFSMS/MVS DFSMSdfp Advanced Services*.

Examples

```
//D3  DD  DSNAME=DSDEF,DISP=(NEW,KEEP),UNIT=3350,
//      VOLUME=SER=668888,SPACE=(TRK,(1,1)),EXPDT=2006/032
//D4  DD  DSNAME=DSFS.PGM,DATACLAS=DCLAS2,DISP=(NEW,KEEP),
//       EXPDT=2006/032
```

Release of Unused Direct Access Space in End Processing

To request that the system release direct access space that was allocated to an output data set but was not used, code:

```
//ddname  DD  SPACE=(TRK,(quantity),RLSE),...
//ddname  DD  SPACE=(CYL,(quantity),RLSE),...
//ddname  DD  SPACE=(blklgth,(quantity),RLSE),...
```

The system releases space only if the data set is open for output and the last operation was a write. The system does not release space if the step terminates abnormally. The system ignores a request to release unused space if:

- Another job is sharing the data set.
- Another task in the same job is processing an OPEN, CLOSE, EOV, or FEOV request for the data set.
- Another data control block is open for the data set.
- The CLOSE macro instruction contains TYPE=T.

Example

```
//DD3  DD  DSNAME=DEPTDS,DISP=(NEW,KEEP),UNIT=DISK,
//        SPACE=(CYL,20,RLSE)
```

Disposition End Processing of Volume

Disposition of the tape or direct access volume containing a data set is controlled by coding:

```
//ddname  DD  VOLUME=(PRIVATE,RETAIN,...),...
//ddname  DD  VOLUME=(PRIVATE,...),...
//ddname  DD  VOLUME=(,RETAIN,...),...
```

RETAIN Support

RETAIN can be specified only for tape.

In a JES3 system, RETAIN is supported only by MVS. If coded on a DD statement for a data set on an MVS-managed tape device, the system designates the volume as retained. If coded on a DD statement for a data set on a JES3-managed tape device, JES3 ignores the RETAIN parameter when issuing KEEP/RETAIN messages and when performing unallocation at the end of the job. However, if RETAIN is coded for a data set on a JES3-managed tape device and the tape volume is to be shared with a later step, JES3 designates the volume as retained.

Disposition of Removable Volumes

If a removable direct access or tape volume is designated as private, the system asks the operator to demount the volume at the end of the step and place it in the installation library.

If a removable direct access or tape volume is designated as public, the system keeps it mounted for other uses, unless the device is needed for another allocation.

Tape Volumes in JES2

When disposing of tape volumes, a JES2 system marks them as follows:

- **Keep (K):** The volume is to be placed in the installation tape library. K is the designation for a private tape volume.
- **Scratch (D):** The volume can be used whenever a DD statement makes a nonspecific request for a tape volume. D is the designation for a public tape volume.

Examples

```
Volumes treated as private, demounted, and kept:

//EX1  DD  DSNAME=A,DISP=(NEW,KEEP),VOLUME=PRIVATE,UNIT=TAPE
//EX2  DD  DSNAME=B,DISP=OLD,VOLUME=SER=223344,UNIT=DISK
//EX3  DD  DSNAME=H,DISP=OLD

Volumes treated as public and kept mounted for other uses:

//EX4  DD  DSNAME=D,UNIT=TAPE
//EX5  DD  DSNAME=&&TEMP,UNIT=DISK
```

Volume Retention

The system designates a tape volume as *retained* (R) if the volume contains one of the following:

- A passed data set
- A data set requested by a DD statement with RETAIN in the VOLUME parameter.

Retained Private Tape Volume

If RETAIN is coded or the data set is passed, the system designates the volume as R, does not demount the mounted volume, and does not rewind the tape when the data set is closed or at the end of the step.

Retained Public Tape Volume

If RETAIN is coded or the data set is passed, the system designates the volume as R, but asks the operator to demount it and keep it near for possible use later.

Use of Retained Volumes

In a multiple step job, if there is a period when a volume is not in use, you can specify RETAIN to try to keep the volume mounted. If the volume remains

mounted, the operator does not have to demount and remount it, and the job does not have to wait until the volume is remounted.

Demounting of Passed or Retained Volumes

Even if you specify RETAIN or a disposition of PASS, the operator can still unload the volume or, if the device is needed for another step in the same or another job, the system can allocate the device and demount the volume. Either can occur when the device on which the volume is mounted is not allocated to the job step that specified RETAIN or, for unlabeled tapes, when the volume requires verification.

Example

```
//EXDD   DD  DSNAME=TAPEDS,DISP=(NEW,CATLG,DELETE),UNIT=3420,
//           VOLUME=(PRIVATE,RETAIN)
```

Note: CLOSE options may cause RETAIN to be overridden. See the discussion of the CLOSE macro in *DFSMS/MVS Macro Instructions for Data Sets*.

Part 5. Tasks for Requesting Sysout Data Set Resources

Chapter 18. Sysout Resources - Identification 18-1
Identification as a Sysout Data Set 18-1
Identification of Output Class 18-2
Identification of Data Set on 3540 Diskette Input/Output Unit 18-2

Chapter 19. Sysout Resources - Description 19-1
Description of Data Attributes 19-1

Chapter 20. Sysout Resources - Protection 20-1
Protection of Printed Output 20-1

Chapter 21. Sysout Resources - Performance Control 21-1
Performance Control by Queue Selection (non-APPC) 21-1

Chapter 22. Sysout Resources - Processing Control 22-1
Processing Control with Additional Parameters 22-1
Adding Parameters from OUTPUT JCL Statement 22-2
Adding Parameters from JES2 /*OUTPUT Statement 22-4
Adding Parameters from JES3 //*FORMAT Statement 22-4
Processing Control by Segmenting 22-5
Processing Control with Other Data Sets 22-5
Using Output Class 22-5
Using Sysout Data Set Size in a JES3 System 22-5
Using Groups in a JES2 System 22-6
Processing Control by External Writer 22-7
Processing Control by Mode 22-7
Processing Control by Holding 22-8
Holding Using the DD Statement 22-8
Holding Using the OUTPUT JCL Statement 22-8
Releasing Held Data Set 22-9
Processing Control by Suppressing Output 22-9
Using Dummy Status to Suppress Output 22-9
Using Class to Suppress Output in a JES2 System 22-10
Using the OUTPUT JCL Statement to Suppress Output in a JES2 System 22-10
Processing Control with Checkpointing 22-10
Processing Control by Print Services Facility 22-11
Identifying a Library to PSF 22-11
Use of User Libraries 22-11
Considerations for Library Data Sets 22-12

Chapter 23. Sysout Resources - End Processing 23-1
Unallocation End Processing 23-1

Chapter 24. Sysout Resources - Destination Control 24-1
Destination Control to Local or Remote Device or to Another Node 24-1
Multiple Destinations 24-2
Controlling Output Destination in a JES2 Network 24-2
Controlling Output Destination in a JES3 Network 24-5
Destination Control to Another Processor in a JES3 System 24-6
Destination Control to Internal Reader 24-6

© Copyright IBM Corp. 1988, 1999

Part 5. Tasks for Requesting Sysout Data Set Resources

Destination Control to Terminal 24-7
Destination Control to Assist in Sysout Distribution 24-8

Chapter 25. Sysout Resources - Output Formatting 25-1
Output Formatting to Any Printer 25-1
Output Formatting to 3800 Printing Subsystem 25-2
Copy Modification 25-3
Character Arrangements 25-3
Output Formatting to 3211 Printer with Indexing Feature in a JES2 System . 25-4
Output Formatting to Punch 25-5
Interpretation of Punched Cards 25-5
Output Formatting of Dumps on 3800 Printing Subsystem 25-5

Chapter 26. Sysout Resources - Output Limiting 26-1
Output Limiting 26-1
Limiting Output in an APPC Scheduling Environment 26-1
Limiting Output in a Non-APPC Scheduling Environment 26-2
Actions when Limit Exceeded 26-2

Chapter 27. Sysout Resources - USERDATA OUTPUT JCL Keyword . . 27-1
References 27-1
Examples 27-1

This part describes how to create system output (sysout) data sets, which are output data sets processed by JES2 or JES3. The task required to request a sysout data set is:

- Identification

Other tasks can optionally be performed:

- Description
- Protection
- Performance control
- Processing control
- End processing
- Output destination
- Output formatting
- Output limiting

Processing Output: The two ways to process output data sets are:

- Define a sysout data set and how it is to be processed and allow the job entry subsystem to process it. JES writes the data set to a spool device. Then JES or an external writer prints or punches it on a local or remote printer or punch, or JES transmits it to a remote output device or node.
- Define an output data set and specify in the DD statement UNIT parameter the device on which the output should be written. The system allocates the device exclusively to the job. Data management routines write the output from the program to the specified device.

This part describes how sysout data sets are defined and processed.

Chapter 18. Sysout Resources - Identification

Figure 18-1. Identification Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Identification					
as a sysout data set	SYSOUT				
name (last qualifier)	DSNAME				
of output class	class on SYSOUT	CLASS	MSGCLASS on JOB with SYSOUT=* or CLASS=* and SYSOUT=(,)		
of data set on 3540 Diskette Input/Output Unit	DSID				

Identification as a Sysout Data Set

To define an output data set as a sysout data set, code:

```
//ddname  DD  SYSOUT=class
//ddname  DD  SYSOUT=(class,writer-name,form-name)
//ddname  DD  SYSOUT=(class,writer-name,code-name)
//ddname  DD  SYSOUT=*
//ddname  DD  SYSOUT=(,)
```

Note that SMS does not manage sysout data sets.

Naming a Sysout Data Set

To assign the last qualifier of the system-generated name for a sysout data set, code the DSNAME parameter with the SYSOUT parameter.

Examples

```
//EX1  DD  SYSOUT=B,DSNAME=&&PRTREC
//EX2  DD  SYSOUT=(A,,FM23)
//EX3  DD  SYSOUT=(F,,CD3),DSNAME=&&PAYOUT
//EX4  DD  SYSOUT=*

//EX5  OUTPUT CLASS=E
//EX6  DD  SYSOUT=(,),OUTPUT=*.EX5
```

© Copyright IBM Corp. 1988, 1999

Identification of Output Class

The installation sets up output classes during JES2 or JES3 initialization. Each class is assigned processing characteristics and is printed or punched on certain devices. The output class for a sysout data set is identified by coding one of the following:

```
//ddname   DD  SYSOUT=class

//jobname  JOB  acct,progname,MSGCLASS=class
//stepname EXEC PGM=x
//ddname   DD  SYSOUT=*
//name     OUTPUT CLASS=class
//ddname   DD   SYSOUT=(,),OUTPUT=*.name
```

For example, the installation could define output class W to contain low-priority output; class Y to contain output to be printed on a special form, so that the JCL would not need to request the form; and class J to be reserved for high-volume output.

To print the sysout data set and messages from the job on the same output listing, see “Printing Job Log and Sysout Data Sets Together” on page 7-8.

Examples

```
//X1    DD   SYSOUT=A

//JOBA  JOB  ,'I. BUTLER',MSGCLASS=B
//ST1   EXEC PGM=ANY
//X2    DD   SYSOUT=*
//OUTA  OUTPUT CLASS=C
//X3    DD   SYSOUT=(,),OUTPUT=*.OUTA
```

Identification of Data Set on 3540 Diskette Input/Output Unit

Data sets are written on 3540 diskette volumes by coding:

```
//ddname  DD  SYSOUT=(class,diskette-writer),DSID=id
//ddname  DD  SYSOUT=(class,diskette-writer),DSID=(id,V)
```

A system command, from the operator or in the input stream, must start the diskette writer before the DD statement is processed.

For more information on the 3540 diskette, see *3540 Programmer's Reference*. For information on external writers, see *OS/390 JES2 Initialization and Tuning Guide* or *OS/390 JES3 Initialization and Tuning Guide*.

Example

```
//EX7  DD  SYSOUT=(W,WRT3540),DSID=MYDS5
```

Chapter 19. Sysout Resources - Description

Figure 19-1. Description Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Description					
of data attributes	DCB				

Description of Data Attributes

Your application program may require the coding of the DCB parameter on the DD statement. And, it might operate differently if the DCB parameter is coded.

If you specify an external writer on the SYSOUT parameter, the external writer may require the DCB parameter on the DD statement that was used to create the data set.

Consult the documentation for your application program or the external writer, if appropriate, for further information about DCB subparameters that may be required or recommended.

Example

```
//OUT3  DD  SYSOUT=(H,WRTPGM),DCB=(RECFM=FB,LRECL=133,BLKSIZE=532)
```

© Copyright IBM Corp. 1988, 1999

Chapter 20. Sysout Resources - Protection

Figure 20-1. Protection Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Protection					
of printed output		DPAGELBL SYSAREA			

Protection of Printed Output

To add security protection to the printed output of sysout data sets, code the following parameters on the OUTPUT JCL statement:

```
//name   OUTPUT  DPAGELBL=YES,SYSAREA=YES
```

Use DPAGELBL=YES to indicate that the system should print the security label on each page of printed output. Use SYSAREA=YES to indicate that the system should reserve an area for the security label on each page of printed output.

The security label represents a security level and categories defined to the Resource Access Control Facility (RACF) by the security administrator at your installation. Use the DPAGELBL and SYSAREA parameters on an OUTPUT JCL statement as instructed by your security administrator.

Example

```
//JOBB   JOB   1,'JIM WOOSTER',SECLABEL=CONF

//PSRPT  OUTPUT  DPAGELBL=YES,SYSAREA=YES,FORMS=TSEC
```

© Copyright IBM Corp. 1988, 1999

Chapter 21. Sysout Resources - Performance Control

Figure 21-1. Performance Control Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Performance Control					
by queue selection		PRTY			

Performance Control by Queue Selection (non-APPC)

The PRTY parameter has no effect in an APPC scheduling environment. If you code PRTY, the system will check it for syntax and ignore it.

You can specify the priority at which the sysout data set enters the output queue by coding:

```
//name OUTPUT PRTY=nnn
```

Use the priority to increase a sysout data set's priority so it will be printed sooner than it otherwise might have been.

Ignoring Priority

The installation can instruct the system to ignore a priority specified on an OUTPUT JCL statement.

Example

```
//OUTA  OUTPUT PRTY=255
//MYDS  DD     SYSOUT=F,OUTPUT=*.OUTA
```

This example requests the highest priority possible.

© Copyright IBM Corp. 1988, 1999

Chapter 22. Sysout Resources - Processing Control

Figure 22-1. Processing Control Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Processing Control					
with additional parameters	OUTPUT code-name on SYSOUT	DEFAULT			
by segmenting	SEGMENT				
with other data sets	class on SYSOUT	THRESHLD (JES3 only) GROUPID (JES2 only)			
by external writer	writer-name on SYSOUT	WRITER			
by mode		PRMODE			
by holding	HOLD class on SYSOUT	CLASS OUTDISP			
by suppressing output	DUMMY class on SYSOUT				
with checkpointing		CKPTLINE CKPTPAGE CKPTSEC			
by Print Services Facility (PSF)		COLORMAP COMSETUP DUPLEX FORMDEF FORMLEN INTRAY OFFSETXB OFFSETXF OFFSETYB OFFSETYF PAGEDEF PRTERROR RESFMT USERLIB			

Processing Control with Additional Parameters

To control how a sysout data set is processed, specify parameters on the DD statement with the SYSOUT parameter. Code the following statements to supply additional parameters:

```
By explicit reference to earlier OUTPUT JCL statement:
     //name   OUTPUT parameters
     //ddname DD     SYSOUT=class,OUTPUT=*.name,parameters

By implicit reference to earlier default OUTPUT JCL statement:
     //name   OUTPUT  DEFAULT=YES,parameters
     //ddname DD      SYSOUT=class,parameters
```

© Copyright IBM Corp. 1988, 1999

Adding Parameters from OUTPUT JCL Statement

JES combines the parameters from the sysout DD statement and one OUTPUT JCL to write the sysout data set. If a parameter appears on both statements, JES uses the parameter from the DD statement.

Note that if an OUTPUT JCL statement contains both JESDS and CLASS parameters, this CLASS will override the MSGCLASS parameter on the JOB statement for the specified JES datasets.

Multiple References

A sysout DD statement can reference more than one OUTPUT JCL statement. For each reference to an OUTPUT JCL statement, JES processes the sysout data set using the parameters of the DD statement combined with the parameters from one of the OUTPUT JCL statements.

Example 1

```
//JOB1    JOB      ,'DEPT. 25'
//OUT1    OUTPUT   COPIES=8,DEST=FRANCE
//OUT2    OUTPUT   COPIES=2,FORMS=A,DEFAULT=YES
//STEP1   EXEC     PGM=DEMENT
//OUT3    OUTPUT   DEFAULT=YES,COPIES=5,DEST=REMULAC
//INPUT   DD       DSN=RHINO
//MFK1    DD       SYSOUT=A
//MFK2    DD       SYSOUT=B,OUTPUT=*.OUT1
```

This example shows an explicit reference to an OUTPUT JCL statement. Note that with an explicit reference, all default OUTPUT JCL statements are ignored.

- The system processes the output from DD statement MFK1 using the options on the OUTPUT statement OUT3 (1) because MFK1 does not contain an OUTPUT parameter and (2) because OUT3 contains DEFAULT=YES and is in the same step as MFK1. MFK1 cannot implicitly reference the job-level default statement OUT2 because of step-level default statement OUT3. If STEP1 had not contained OUT3, MFK1 would have referenced statement OUT2.
- The system processes the output from DD statement MFK2 according to the processing options on the job-level OUTPUT JCL statement OUT1 because DD statement MFK2 explicitly references OUT1 using the OUTPUT parameter. Note that the system ignores the processing options on all default OUTPUT JCL statements (OUT2 and OUT3).

Example 2

```
//EXAMP  JOB     MSGCLASS=A
//OUT1   OUTPUT  DEFAULT=YES,DEST=COMPLEX7,FORMS=BILLING,
//           CHARS=(AOA,AOB),COPIES=2
//OUT2   OUTPUT  DEFAULT=YES,DEST=COMPLEX1
//STEP1  EXEC    PGM=ORDERS
//R1     DD      SYSOUT=A
//R2     DD      SYSOUT=A
//STEP2  EXEC    PGM=BILLING
//OUT3   OUTPUT  DEFAULT=YES,DEST=COMPLEX3
//B1     DD      SYSOUT=A
//B2     DD      SYSOUT=A,OUTPUT=(*.OUT3,*.OUT2)
//STEP3  EXEC    PGM=REPORTS
//OUT4   OUTPUT  FORMS=SHORT,DEST=COMPLEX1
//RP1    DD      SYSOUT=A
//RP2    DD      SYSOUT=A,OUTPUT=(*.STEP2.OUT3,*.OUT1)
//
```

This example shows how the position of the OUTPUT JCL statement affects the processing of the sysout data sets.

In STEP1, the system processes DD statements R1 and R2 using the processing options specified on job-level OUTPUT JCL statements OUT1 and OUT2 because

- DEFAULT=YES is specified on OUTPUT JCL statements OUT1 and OUT2, and
- there is no OUTPUT JCL statement with DEFAULT=YES within STEP1.
- The OUTPUT parameter is not specified on DD statements R1 and R2.

In STEP2, the system processes DD statement B1 using the processing options specified on OUTPUT JCL statement OUT3 because:

- DEFAULT=YES is specified on OUTPUT JCL statement OUT3 and OUTPUT JCL statement OUT3 is within the job step STEP2.
- The OUTPUT parameter is not specified on DD statement B1.
- OUTPUT JCL statement OUT3 is within STEP2; therefore, the system ignores the DEFAULT=YES specification on job-level OUTPUT JCL statements OUT1 and OUT2 when processing DD statement B1.

In STEP2, the system processes DD statement B2 using the processing options specified on OUTPUT JCL statements OUT3 and OUT2 because:

- Both of the OUTPUT JCL statements are explicitly referenced from the SYSOUT statement. Explicitly-referenced OUTPUT JCL statements can be in any previous procedure or step, before the DD statement in the current step, or at the job-level.
- Note that default OUTPUT JCL statement OUT1 is ignored when processing the data set defined by DD statement B2 because B2 explicitly references OUTPUT JCL statements OUT3 and OUT2.

In STEP3, the system processes DD statement RP1 using the output processing options specified on the job-level OUTPUT JCL statements OUT1 and OUT2 because:

- DEFAULT=YES is specified on OUTPUT JCL statements OUT1 and OUT2, and
- no OUTPUT JCL statement with DEFAULT=YES is coded within STEP3.

- The OUTPUT parameter is not specified on DD statement RP1.

Note: In STEP3, OUTPUT JCL statement OUT4 is not used at all because it does not have DEFAULT=YES coded, and no DD statement explicitly references OUT4.

In STEP3, DD statement RP2 is processed using OUTPUT statements OUT3 and OUT1. You can explicitly reference an OUTPUT JCL statement in another step if you use a fully qualified reference, such as the reference to OUTPUT statement OUT3 used on DD statement RP2.

You may explicitly reference an OUTPUT JCL statement with DEFAULT=YES coded, such as the reference to OUT1 from DD statement RP2. The system ignores the DEFAULT parameter and uses the remaining processing options according to the normal rules that apply when coding explicit references.

Example 3

```
//STEP1  EXEC   PGM=MFK
//OUT1   OUTPUT COPIES=6,DEST=NY,FORMS=BILLS
//OUT2   OUTPUT COPIES=2,DEST=KY,FORMS=LOG
//REF1   DD     SYSOUT=A,OUTPUT=(*.OUT1,*.OUT2)
```

In the example, two sets of output are created from DD statement REF1. One of the sets will go to NY and have six copies printed on the form defined as BILLS. The other set will go to KY and have two copies printed on the form defined as LOG.

Adding Parameters from JES2 /*OUTPUT Statement

JES2 can combine the parameters from the sysout DD statement and a referenced /*OUTPUT statement to write the sysout data set.

Because the OUTPUT JCL statement provides greater output processing capabilities, an installation should consider changing its /*OUTPUT statements to OUTPUT JCL statements.

Be careful when doing the change. Before the change, the third subparameter in the DD SYSOUT parameter references a JES2 /*OUTPUT statement. But, if the DD statement references an OUTPUT JCL statement, the system interprets the third subparameter as the name of forms to be used in processing the sysout data set.

Adding Parameters from JES3 //*FORMAT Statement

A JES3 //*FORMAT statement can explicitly reference a sysout DD statement to make JES3 combine the parameters from the sysout DD statement and the //*FORMAT statement to write the sysout data set.

Because the OUTPUT JCL statement provides greater output processing capabilities, an installation should consider changing its //*FORMAT statements to OUTPUT JCL statements.

Processing Control by Segmenting

To control the size of a sysout data set segment, code the SEGMENT parameter on a sysout DD statement. SEGMENT is supported in JES2 systems only.

When you code SEGMENT, you determine the number of logical line-mode pages to be written to a sysout data set. This allows you to print part of the output while a job is still executing, or to use multiple printers to print multiple segments.

```
//DD1    DD  SYSOUT=A,SEGMENT=200
```

In this example, when the system writes 200 pages to a sysout data set, the segment is spun and a new segment is allocated.

Processing Control with Other Data Sets

Using Output Class

JES prints on the same output listing the output from all sysout data sets for a job if the class, forms, FCB, UCS, and DEST parameters are the same and if an external writer is not specified. The installation can choose to print all sysout data sets that specify the same output class as the JOB statement MSGCLASS parameter on the same listing, even though the forms, FCB, UCS, and sometimes the DEST parameters are different.

Example 1

```
//DD1    DD  SYSOUT=(C,,FM34)
//DD2    DD  SYSOUT=(C,,FM34)
```

The sysout data sets for DD1 and DD2 are written on the same output listing.

Example 2

```
//JEX  JOB  ,'M. BIRDSALL',MSGCLASS=D
//ST1  EXEC PGM=WKRPT
//DDA  DD   SYSOUT=*
//DDB  DD   SYSOUT=D
```

The sysout data sets for DDA and DDB are written on the same output listing as the job log.

Using Sysout Data Set Size in a JES3 System

To control whether all the sysout data sets in one class from a job are printed together or as separate units of work, code one of the following groups:

```
//name    OUTPUT THRESHLD=limit
//ddname1 DD     SYSOUT=class,OUTPUT=*.name
//ddname2 DD     SYSOUT=class,OUTPUT=*.name

//name    OUTPUT DEFAULT=YES,CLASS=class,THRESHLD=limit
//ddname1 DD     SYSOUT=(,)
//ddname2 DD     SYSOUT=(,)
```

JES3 calculates the size of the sysout data set(s) as the number of records multiplied by the number of copies requested. When the size exceeds the

THRESHLD value, JES3 creates a new unit of work on a data set boundary, and queues it for printing.

Use of THRESHLD

If a sysout data set or all the sysout data sets in the same class from a job are large, or large numbers of copies are requested, the THRESHLD limit can be used to print copies simultaneously by different printers.

Examples

```
//OUTA  OUTPUT THRESHLD=10000
//MYDS1 DD     SYSOUT=C,OUTPUT=*.OUTA,COPIES=5
//GRDS  DD     SYSOUT=C,OUTPUT=*.OUTA,COPIES=3

//OUTB  OUTPUT DEFAULT=YES,CLASS=C,THRESHLD=10000
//MYDS1 DD     SYSOUT=(,),COPIES=5
//GRDS  DD     SYSOUT=(,),COPIES=3
```

Using Groups in a JES2 System

In JES2 systems, you can group sysout data sets together by coding:

```
//name  OUTPUT GROUPID=output-group
```

Sysout data sets in the same group are processed together in the same location and time.

Subgroups

You can always group sysout data sets with similar processing characteristics. But, you cannot group sysout data sets with different output classes, destinations, processing modes (PRMODE), writer names, or groupids. If you use GROUPID to group dissimilar data set, the system breaks down the group into subgroups of sysout data sets with identical classes, destinations, processing modes, writer names, and groupids.

Demand Setup Groups

The installation controls whether a group can contain sysout data sets with different printer setup requirements, such as forms. Such groups are called demand setup groups. If demand setup grouping is not permitted, data sets with different setup requirements are placed in different subgroups.

Example

```
//TEST1  JOB     MSGCLASS=B
//OUT1   OUTPUT  GROUPID=GRP10,UCS=PN,DEST=RT6,DEFAULT=YES
//STEP1  EXEC    PGM=REPORT
//RP1    DD      SYSOUT=A
//RP2    DD      SYSOUT=B
//RP3    DD      SYSOUT=A
```

In this example, two subgroups are created for the three sysout data sets because of the different output classes. One subgroup contains data sets RP1 and RP3; the other contains RP2.

Processing Control by External Writer

To request that a sysout data set be processed by an IBM-supplied or user-written external writer, rather than the installation's JES, code one of the following:

```
//ddname  DD  SYSOUT=(class,writer-name)

//name    OUTPUT WRITER=writer-name
//ddname  DD     SYSOUT=class,OUTPUT=*.name


//name    OUTPUT DEFAULT=YES,WRITER=writer-name
//ddname  DD     SYSOUT=class
```

For an external writer, the operator determines which sysout data sets are selected. This can cause certain data sets to be printed on the same listing even though all of the forms, FCB, UCS, and DEST parameters are not the same. The operator must start the external writer for a sysout data set to be printed or punched.

For more information on external writers, see *OS/390 JES2 Initialization and Tuning Guide* or *OS/390 JES3 Initialization and Tuning Guide*.

Examples

```
//DS1  DD     SYSOUT=(H,MYWRIT)

//OTA  OUTPUT WRITER=MYWRIT
//DS1  DD     SYSOUT=H,OUTPUT=*.OTA


//OTB   OUTPUT DEFAULT=YES,WRITER=MYWRIT
//DS1   DD     SYSOUT=H
```

Processing Control by Mode

To request the correct process mode for a sysout data set, code one of the following:

```
//name  OUTPUT PRMODE=LINE
//name  OUTPUT PRMODE=PAGE
//name  OUTPUT PRMODE=process-mode
```

JES schedules the sysout data set to a printer that can operate in the specified mode.

Examples

```
//OTS  OUTPUT PRMODE=PAGE
//ABC  DD     SYSOUT=F,OUTPUT=*.OTS
```

JES schedules data set ABC to a 3800 Printing Subsystem Model 3, which can print in page mode. Output class F must handle processing for a 3800 model 3.

Processing Control by Holding

Some of the reasons for holding a data set are:

- To make it available for inspection from a time-sharing terminal.
- If it is very large, to prevent it from monopolizing an output device until smaller data sets are written.
- If it requires special forms, to delay its printing or punching until the operator can supply the forms.

Holding Using the DD Statement

To hold a sysout data set on the JES spool and delay its printing or punching, code one of the following:

```
//ddname  DD  SYSOUT=class,HOLD=YES

 Or where the specified class is designated
 as a held class during JES initialization:

//ddname  DD  SYSOUT=class

//name    OUTPUT CLASS=class
//ddname  DD     SYSOUT=(,),OUTPUT=*.name

//name    OUTPUT DEFAULT=YES,CLASS=class
//ddname  DD     SYSOUT=(,)
```

The HOLD parameter overrides any disposition specified on the OUTDISP parameter.

Holding Using the OUTPUT JCL Statement

Use the OUTDISP parameter with JES2 only.

You can code a sysout data set disposition that is based on the success of the job. The OUTDISP parameter of the OUTPUT JCL statement allows you to specify a normal sysout disposition and an abnormal sysout disposition. Note that the OUTDISP abnormal sysout disposition is not supported in an APPC scheduling environment. The system uses the normal disposition when the job completes successfully. It uses the abnormal disposition when the job does not complete successfully, due to a JCL error, an abend, or job termination resulting from a condition code.

For example, the following statement will cause the system to hold a sysout data set when the job completes normally or abnormally.

```
//HELDDS    OUTPUT  OUTDISP=(HOLD,HOLD)
```

Coding OUTDISP=(HOLD,HOLD) is equivalent to coding HOLD=YES on the DD statement.

The OUTDISP parameter allows you to specify the following dispositions for a sysout data set:

- HOLD allows the system to hold a sysout data set. When the user or operator releases the data set, the system prints and then purges it.

- WRITE allows you to print a sysout data set and purge it after it is printed.
- KEEP allows you to print and keep the sysout data set. After it is printed, the disposition changes to LEAVE.
- LEAVE allows the system to hold a sysout data set until the user or operator releases it. When the sysout data set is released, the disposition changes to KEEP.
- PURGE allows you to delete a sysout data set without printing it.

Releasing Held Data Set

When a data set is to be held, JES places the sysout data set on a hold queue until the operator releases it. The system issues no message to tell the operator that the data set is being held. Therefore, when the data set can be processed, ask the operator to release it or release it from a TSO/E userid with a TSO/E OUTPUT command. See *OS/390 TSO/E Command Reference* for information on TSO/E commands.

Examples

```
//DD1  DD  SYSOUT=C,HOLD=YES

//DD2  DD  SYSOUT=J

//OT1  OUTPUT CLASS=J
//DD3  DD     SYSOUT=(,),OUTPUT=*.OT1

//OT2  OUTPUT DEFAULT=YES,CLASS=J
//DD4  DD     SYSOUT=(,)
```

In all these examples, the installation defined class J as a held class during JES initialization.

Processing Control by Suppressing Output

Using Dummy Status to Suppress Output

If you want to suppress processing of a sysout data set, assign it a dummy status by coding:

```
//ddname  DD  DUMMY,SYSOUT=class,...
```

Effect of Dummy Sysout Data Set

When DUMMY is coded, the system ignores the SYSOUT parameter and bypasses all output operations to the spool. The sysout data set is not printed or punched.

Use of a Dummy Sysout Data Set

Defining a sysout data set as a dummy data set is useful when testing a program; you do not want data sets printed until you are sure they contain meaningful output.

Nullifying a Dummy Sysout Data Set

When the sysout data set is to be processed, remove the DUMMY parameter from the sysout DD statement.

Examples

```
//EXA  DD  DUMMY,SYSOUT=A
//EXB  DD  DUMMY,SYSOUT=(B,WRT),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
```

Using Class to Suppress Output in a JES2 System

To suppress the printing or punching of a sysout data set in a JES2 system, code one of the following:

```
//ddname  DD  SYSOUT=class

//name    OUTPUT CLASS=class
//ddname  DD     SYSOUT=(,),OUTPUT=*.name

//name    OUTPUT DEFAULT=YES,CLASS=class
//ddname  DD     SYSOUT=(,)
```

During JES2 initialization, the installation must specify that the requested class contains data sets that are deleted before being printed or punched.

Use of Output Suppression

Use this technique to suppress the output of started tasks.

Examples

```
//DD2  DD  SYSOUT=S

//OT1  OUTPUT CLASS=S
//DD3  DD     SYSOUT=(,),OUTPUT=*.OT1

//OT2  OUTPUT DEFAULT=YES,CLASS=S
//DD4  DD     SYSOUT=(,)
```

In all these examples, the installation defined class S as an output suppression class.

Using the OUTPUT JCL Statement to Suppress Output in a JES2 System

By coding the PURGE subparameter of the OUTDISP parameter, you can keep a sysout data set from printing.

Example

```
//NOPRT   OUTPUT  OUTDISP=(PURGE,PURGE)
```

Processing Control with Checkpointing

To write a checkpoint while JES is processing a sysout data set, code one of the following:

```
//name     OUTPUT CKPTLINE=number,CKPTPAGE=number
//ddname   DD     SYSOUT=class
              .
              .
//name     OUTPUT CKPTSEC=number
//ddname   DD     SYSOUT=class
```

Example 1

```
//J2       JOB    ,MHB
//S1       EXEC   PGM=ABC
//OT2      OUTPUT CKPTLINE=60,CKPTPAGE=40
//DDB      DD     SYSOUT=C
```

JES writes a checkpoint every 40 logical pages. A logical page contains 60 lines.

Example 2

```
//J2       JOB    ,MHB
//S1       EXEC   PGM=DEF
//OT2      OUTPUT CKPTSEC=60
//DDB      DD     SYSOUT=D
```

JES writes a checkpoint every 60 seconds.

Processing Control by Print Services Facility

To control how the Print Services Facility (PSF) prints a sysout data set on a page-mode printer (such as a 3800 Printing Subsystem Model 3), code:

```
//name   OUTPUT PAGEDEF=membername,FORMDEF=membername
//ddname DD     SYSOUT=class,OUTPUT=*.name
```

The PAGEDEF and FORMDEF parameters identify library members in the library named in the cataloged procedure used to initialize the PSF, or in a library specified on the USERLIB parameter of the OUTPUT JCL statement. These members contain statements that specify how PSF is to process the sysout data set.

Examples

```
//OTPSF OUTPUT  PAGEDEF=PSLONG,FORMDEF=FSBILL
//MYPNT DD      SYSOUT=N,OUTPUT=*.OTPSF
```

To control how PSF prints a sysout data set on a microfilm device, code:

```
//name   OUTPUT  COMSETUP=H1SETUP
```

The COMSETUP parameter specifies the name of a microfile setup resource that contains setup information for FSS microfilm devices.

Identifying a Library to PSF

The USERLIB parameter of the OUTPUT JCL statement identifies a library containing AFP resources to PSF. Libraries specified on USERLIB are concatenated to system libraries, and PSF checks them before the system libraries for the requested resources.

Use of User Libraries

USERLIB allows you to maintain copies of AFP resources that are not accessible to all users. This enables you to:

- Maintain copies of secure resources, such as signatures, in private data sets
- Keep resources that are being tested in a private data set during the testing period

- Personalize and maintain your own library.

The USERLIB parameter is supported for deferred-printing mode. It is not supported for direct-printing mode.

Considerations for Library Data Sets

PSF dynamically allocates libraries that you specify on the USERLIB parameter. The library is allocated to the PSF address space, with a shared data set disposition. After processing the sysout data set, PSF dynamically unallocates the library. When planning to use the USERLIB parameter, take into account the dynamic allocation system constraints on performance.

Requirements

The following are requirements for the libraries.

- If RACF is installed on your system, the job submitter must have RACF read access to the libraries specified on USERLIB.
- The libraries must be cataloged in a catalog available to PSF/MVS.
- The libraries must be accessible to PSF during its processing of the sysout data set. Note that the time of processing might be days after job submission, and processing might occur on a node other than that which submitted the job.

See *PSF/MVS Application Programming Guide* for more information on USERLIB.

Chapter 23. Sysout Resources - End Processing

Figure 23-1. End Processing Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
End processing					
unallocation	FREE SPIN				

Unallocation End Processing

Normally JES2 or JES3 schedules all sysout data sets from a job for printing or punching when all the system-managed data sets are processed at the end of the job.

Spinning Data Sets

Sysout data sets can be scheduled for printing or punching when the data set is closed before the job completes execution. Code:

```
//ddname  DD  SYSOUT=class,FREE=CLOSE
```

These data sets are called **spin data sets**.

If the step continues processing after the close, the sysout data set may be printed concurrently with the last of the step's execution.

Use of Spinning

Use FREE=CLOSE to let JES begin printing or punching a sysout data set before a long job step is finished.

Example

```
//STEP1 EXEC PGM=VERYLONG
//SHORT DD   SYSOUT=C,FREE=CLOSE
```

To make a sysout data set available for printing immediately, code SPIN=UNALLOC on the sysout DD statement, and dynamically unallocate the data set. Dynamic unallocation can be explicit or through FREE=CLOSE.

© Copyright IBM Corp. 1988, 1999

Chapter 24. Sysout Resources - Destination Control

Figure 24-1. Destination Control Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Destination Control					
to local or remote device or to another node	DEST class on SYSOUT	DEST COMPACT		/*ROUTE PRINT /*ROUTE PUNCH	ORG on //*MAIN
to another processor					ACMAIN on //*MAIN
to internal reader	INTRDR as writer-name on SYSOUT		/*EOF /*DEL /*PURGE /*SCAN		
to terminal	TERM				
to assist in sysout distribution		ADDRESS BUILDING DEPT NAME ROOM TITLE		DEST on /*OUTPUT	

Destination Control to Local or Remote Device or to Another Node

To send a sysout data set to a local or remote device or to another node, code one of the following:

```
     //ddname   DD      SYSOUT=class,DEST=destination

     //name     OUTPUT  DEST=destination,COMPACT=compaction-table-name
     //ddname   DD      SYSOUT=class,OUTPUT=*.name
```

```
In a JES2 system:
     /*ROUTE    PRINT   destination
     //ddname   DD      SYSOUT=class

     /*ROUTE    PUNCH   destination
     //ddname   DD      SYSOUT=class
```

```
In a JES3 system, to send to group, node, or remote work station:
     //jobname  JOB     acct,progname
     //*MAIN    ORG=group-or-node.remote
     //stepname EXEC    PGM=x
     //ddname   DD      SYSOUT=class
```

© Copyright IBM Corp. 1988, 1999

Multiple Destinations

For example, to print a report in Chicago, New York, Paris, and Los Angeles, code and reference four OUTPUT JCL statements. Specify a different destination on each; you can code only one destination on each OUTPUT JCL statement.

By referencing OUTPUT JCL statements, you can specify 128 different destinations for a single sysout data set. In addition, you can use each OUTPUT JCL statement to specify processing options for each destination.

Keep in mind that, if a JCL syntax error occurs, the system will ignore the OUTPUT JCL statement and the output will not reach its destination.

Controlling Output Destination in a JES2 Network

In a network, you can route sysout data sets from any node or work station to any node or work station.

Unless overridden by the operator or directed by a destination parameter, a sysout data set is printed or punched at the submitting location. To route a sysout data set to another location, use the following:

WRITER parameter on OUTPUT JCL statement
Specifies an external writer for the sysout data set being defined.

WRITER-NAME subparameter on DD SYSOUT statement
Specifies an external writer for the sysout data set being defined.

Note: The WRITER-NAME subparameter on a DD statement overrides an OUTPUT JCL WRITER parameter.

Electronic Mail and External Writer Processing
Processing for the WRITER ID parameter and USERID parameter for sysout data sets is different with version 4 JES2. Destination userids are not external writer IDs. Processes which select output based on WRITER ID (such as external writer programs) will use the value specified on the WRITER ID parameter when selecting sysout. Processes which select output based on DESTINATION USERID (such as TSO/E RECEIVE) will use the value specified on the DEST parameter when selecting sysout.

- A TSO/E user can issue the TSO/E command RECEIVE and obtain electronic mail if:
  - Sysout userid matches TSO/E userid and the sysout WRITER ID is not specified.
  - Sysout userid and sysout WRITER ID match TSO/E userid
  - Sysout userid Matches TSO/E userid and sysout WRITER ID does not match his TSO/E userid.
- A TSO/E user cannot issue the TSO/E command RECEIVE and obtain electronic mail if:
  - Sysout WRITER ID matches TSO/E userid and sysout userid does not match his TSO/E userid.
  - Sysout WRITER ID matches TSO/E userid and sysout userid is not specified.
- An external writer program can process sysout if:

  - Sysout WRITER ID matches external writer program name and sysout userid is not specified.
  - Sysout WRITER ID and sysout userid match external writer program name.
  - Sysout WRITER ID matches external writer program name and TSO/E userid does not.

- An external writer program cannot process sysout if:
  - Sysout WRITER ID is not specified and external writer program name matches sysout userid.
  - Sysout WRITER ID is specified and does not match external writer program name and sysout userid matches external writer program name.

Networking Considerations

On destination node for output received by NJE (including spool offload):

- If both external writer and destination userid are specified, and both are identical, JES2 will blank out the WRITER ID field during network processing. In this case, a TSO/E user can issue a RECEIVE command and process the sysout as electronic mail. An external writer program cannot process the sysout.
- If both external writer and destination userid are specified, and both are different, the destination userid and the WRITER ID are processed as specified in the JCL. Either a TSO/E destination userid or an external writer program can process the sysout.
- If destination userid only is specified, external WRITER ID is not filled in. A TSO/E user can do a RECEIVE command if his userid matches the destination userid. An external writer program cannot process the sysout.
  - TSO/E user 'CARNEY' can receive - userid matches WRITER ID

    ```
    //CHRISBX   JOB..............
    //CLW       OUTPUT DEST=DB2.CARNEY
    //STEP1     EXEC  PGM=IEBGENER
    //SYSPRINT  DD SYSOUT=(A,CARNEY),OUTPUT=(*.CLW)
    ```

    The $LJ,ALL command will show this as:

    ```
              DEST=CARNEY
                 W=CARNEY
    ```

  - TSO/E user 'MWAI' cannot receive - TSO/E userid does not match sysout userid (even though WRITER ID does):

    ```
    //EGGBERTX   JOB.............
    //TJW        OUTPUT DEST=PLPSC.EGGBERT
    //STEP1      EXEC PGM=.......
    //SYSPRINT   DD SYSOUT=(A,MWAI),OUTPUT=(*.TJW)
    ```

    The $LJ,ALL command will show this as:

    ```
              DEST=EGGBERT
                 W=MWAI
    ```

  - TSO/E user 'BERNER' can receive - TSO/E userid matches in an NJE sysout case - job executes on non-local node:

```
//BERNERX    JOB.............
//ROUTE      XEQ SNJMAS3
//DXP        OUTPUT DEST=PLPSC.BERNER
//STEP1      EXEC PGM=.......
//SYSPRINT   DD SYSOUT=(A,BERNER),OUTPUT=(*.DXP)
```

The $LJ,ALL command will show this as:

```
          DEST=BERNER
            W=(none)
```

Note: External WRITER ID is discarded in NJE sysout processing.

DEST parameter on DD SYSOUT statement
Specifies the destination for the sysout data set being defined.

class subparameter in SYSOUT parameter on DD statement
Specifies the destination for the sysout data set being defined. During JES2 initialization, a destination must have been defined for the requested class.

DEST parameter on OUTPUT JCL statement
Specifies the destination for all referencing sysout data sets.

DEST parameter on /*ROUTE PRINT or PUNCH statement
Specifies the destination of a job's sysout data sets for any node or any remote work station. All sysout data sets that have no specific destination go to the destination in the /*ROUTE statement.

Note: If you send a job to execute and the job has a ROUTE PRINT RMTnnn statement or a ROUTE PRINT Unnnn statement, JES2 returns the output to RMTnnn or Unnnn at the node of origin. For JES2 to print the output at RMTnnn at the executing node, code DEST=NnnnRmmm on an OUTPUT JCL statement or sysout DD statement.

Default Output Destination
If the destination for a data set is stated specifically on the /*OUTPUT control statement, or the JCL OUTPUT or DD statements, the specified destination is used. However, data sets routed to a remote terminal cannot be controlled by the remote operator. Such data sets are owned by the location specified as the default for the job.

For data sets with no destination specified, the default destination is determined by the device from which the job entered the system.

In the case of an internal reader, the DEST parameter for the internal reader allocation determines the default destination. If the DEST parameter is not specified, the default destination for the output is the location at which the job was originally submitted. For example, a job submitted on NODEA can be routed to NODEB for execution; however, the output is returned to NODEA unless the DEST parameter was specified as NODEB or some other location.

Examples

```
//DDFAR1 DD     SYSOUT=E,DEST=NYC

//DDFAR2 DD     SYSOUT=F

//OTFAR  OUTPUT DEST=NYC,COMPACT=TABCM
//DD1    DD     SYSOUT=E,OUTPUT=*.OTFAR

/*ROUTE  PRINT  NYC
//DD3    DD     SYSOUT=E

/*ROUTE  PUNCH  NYC
//DD4    DD     SYSOUT=P
```

For the second example, output class F must be defined during JES2 initialization as having a destination, for example, a node in Los Angeles.

Controlling Output Destination in a JES3 Network

In a network, you can route sysout data sets from any node or work station to any node or work station.

A sysout data set is printed or punched at the submitting location unless:

- The job was submitted from TSO and routed to the NJE network for execution. Unless overridden by the //*MAIN ORG parameter or directed by a destination parameter, the sysout data set will be routed to the node from which the job was submitted and the destination ANYLOCAL.
- The sysout data set destination was changed by the ORG parameter on the MAIN statement, or by a destination parameter.

To route a sysout data set to another location, use the following:

DEST parameter on DD SYSOUT Statement
Specifies the destination for the sysout data set being defined.

DEST parameter on OUTPUT JCL Statement
Specifies the destination for all referencing sysout data sets.

ORG parameter on //*MAIN Statement
Specifies an origin group, network node, or remote work station for the job's sysout data sets.

Output Destination when Remote Job Processing in JES3

For jobs from remote work stations submitted through remote job processing (RJP), the sysout data sets are returned to the originating work station unless another destination is requested in a //*MAIN statement with an ORG parameter, OUTPUT JCL statement, or DD statement.

Examples

```
//DDFAR   DD      SYSOUT=E,DEST=NYC

//OTFAR   OUTPUT  DEST=NYC,COMPACT=TABCM
//DD1     DD      SYSOUT=E,OUTPUT=*.OTFAR

//JEX3    JOB     ,'MAIL A60'
//*MAIN   ORG=NYC
//S3      EXEC    PGM=GHI
//DD4     DD      SYSOUT=E
```

Destination Control to Another Processor in a JES3 System

To direct all of a job's sysout data sets to a TSO/E userid on another processor, code:

```
//*MAIN ACMAIN=processor-id,USER=userid
```

Example

```
//J1   JOB   ,MHB
//*MAIN  ACMAIN=2,USER=D17MHB
//S1   EXEC  PGM=PROG67
//DDA  DD    SYSOUT=G
```

Destination Control to Internal Reader

To make a sysout data set from a job step be a new job, direct the data set to the internal reader. The input to the internal reader must be the JCL statements to run the later job. Code:

```
//ddname DD  SYSOUT=(class,INTRDR)
```

INTRDR is an IBM-reserved name identifying the internal reader. The system places the output records for the internal reader into a buffer in your address space. When this buffer is full, JES places the contents on the spool; later, JES retrieves the new job from the spool.

Message Class for Internal Reader Job

The output class in the SYSOUT parameter becomes the default message class for the job going into the internal reader, unless you code the MSGCLASS parameter on the JOB statement.

Limiting Records to Internal Reader

Use the OUTLIM parameter on the DD statement to limit the number of logical records written to the internal reader.

Sending Internal Reader Buffer Directly to JES

Instead of waiting for the buffer in your address space to fill up, send the contents of the internal reader buffer directly to JES by coding as the last record in the job:

/*EOF
This control statement delimits the job in the data set and makes it eligible for immediate processing.

/*DEL

This control statement cancels the job in the data set and schedules it for immediate output processing. The output consists of any JCL submitted, followed by a message indicating that the job was deleted before execution.

/*PURGE

For JES2 only, this control statement cancels the job in the data set and schedules it for purge processing; no output is produced for the job.

/*SCAN

For JES2 only, this control statement requests that JES2 only scan the job in the data set for JCL errors. The job is not to be executed.

References

For more information on the internal reader, see *OS/390 MVS Programming: Assembler Services Guide*.

Example

```
//JOBA      JOB   D58JTH,HIGGIE
//GENER     EXEC  PGM=IEBGENER
//SYSIN     DD    DUMMY
//SYSPRINT  DD    SYSOUT=A,DEST=NODE1
//SYSUT2    DD    SYSOUT=(M,INTRDR)
//SYSUT1    DD    DATA
//JOBB      JOB   D58JTH,HIGGIE,MSGLEVEL=(1,1)
//REPORTA   EXEC  PGM=SUMMARY
//OUTDD1    DD    SYSOUT=*
//INPUT     DD    DSN=REPRTSUM,DISP=OLD
//JOBC      JOB   D58JTH,HIGGIE,MSGLEVEL=(1,1)
//REPORTB   EXEC  PGM=SUMMARY
//OUTDD2    DD    SYSOUT=A,DEST=NODE2
//INPUT     DD    DSN=REPRTDAT,DISP=OLD
/*EOF
```

- JOBA executes program IEBGENER.
- Program IEBGENER reads JOBB and JOBC from in-stream data set SYSUT1 and writes them to sysout data set SYSUT2, which is submitted to the internal reader.
- The message class for JOBB and JOBC is M, the SYSOUT class specified on DD statement SYSUT2.
- The message class for sysout data set OUTDD1 is M because SYSOUT=* is coded.
- The /*EOF statement specifies that the preceding jobs are to be sent immediately to JES for input processing.

Destination Control to Terminal

To indicate that a sysout data set is going to a terminal for a TSO/E user, code:

```
//ddname  DD  TERM=TS
```

In a batch job, TERM=TS is treated as though SYSOUT=* were coded. For an output data set in a foreground job, TERM=TS specifies that the data set is to be sent to the TSO/E userid.

Example

```
//DD1  DD  TERM=TS
```

Destination Control to Assist in Sysout Distribution

The following OUTPUT JCL parameters print the specified values on the separator pages of output for a sysout data set. An installation can use this information to assist in sysout distribution.

```
//OUTDS   OUTPUT  ADDRESS=delivery-address,BUILDING=building-id,
//          DEPT=department,NAME=preferred-name,ROOM=room,TITLE=report-title
```

The system prints the values for each parameter on sections of the separator pages reserved for each parameter.

Chapter 25. Sysout Resources - Output Formatting

Figure 25-1. Output Formatting Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Output formatting					
to any printer	COPIES FCB form-name on SYSOUT UCS	COPIES FCB FORMS LINECT (JES2 only) UCS CONTROL	forms, copies, and linect on JOB JES2 accounting information	COPIES, FORMS, and LINECT on /*JOBPARM	
to 3800 Printing Subsystem in addition to most of printer parameters	BURST CHARS FLASH MODIFY DCB=OPTCD=J	BURST CHARS FLASH MODIFY TRC		BURST on /*JOBPARM	
to 3211 Printer with indexing feature		INDEX (JES2 LINDEX only)			
to punch	COPIES FCB form-name on SYSOUT DCB=FUNC=I	COPIES FCB FORMS			
of dumps on 3800 Printing Subsystem	CHARS=DUMP FCB=STD3	CHARS=DUMP FCB=STD3			

Output Formatting to Any Printer

To control the formatting of a printed sysout data set, code combinations of the following:

```
In a JES2 or JES3 system:
     //ddname  DD  SYSOUT=(class,writer-name,form-name),COPIES=number,
     //          FCB=fcb-name,UCS=character-set-code

     //name    OUTPUT CONTROL=spacing,COPIES=number,FCB=fcb-name,
     //          FORMS=form-name,UCS=character-set-code

In a JES2 system:
     //name    OUTPUT LINECT=number

     //jobname JOB    (,,,,,forms,copies,,linect)

     /*JOBPARM COPIES=number,FORMS=form-name,LINECT=number
```

Most of the formatting parameters can be coded on several statements. If coded more than once for a sysout data set, JES selects one parameter according to override rules and uses it.

© Copyright IBM Corp. 1988, 1999

Parameters coded on the JOB statement or /*JOBPARM statement apply to all the sysout data sets in the job.

3203 Printer Model 5 in a JES2 System

JES2 treats the 3203 Model 5 the same as a 3211 Printer with the following exceptions:

- The universal character sets, specified in UCS parameters, for the 3203 Model 5 are the same as for the 1403 printer.
- The 3203 Model 5 does not support indexing; therefore, INDEX and LINDEX parameters are ignored.
- The installation cannot explicitly identify the 3203 Model 5 printer to JES2 during JES2 initialization. MVS passes the 3203 Model 5 identification to JES2 through the unit control block (UCB).

For further information on UCS and UCB, see *DFSMS/MVS DFSMSdfp Advanced Services*.

Example 1

```
//DD1     DD  SYSOUT=(A,FMS3),COPIES=5,
//          FCB=IMG7,UCS=AN

//OTA     OUTPUT CONTROL=DOUBLE,COPIES=5,FCB=IMG7,
//          FORMS=FMS3,UCS=AN
```

Use these parameters in any system.

Example 2

```
//OTB     OUTPUT LINECT=60

//J1      JOB    (,,,,,FMS3,5,,60)

/*JOBPARM COPIES=5,FORMS=FMS3,LINECT=60
```

Use these parameters only in a JES2 system.

Output Formatting to 3800 Printing Subsystem

To control the formatting of a sysout data set printed on a 3800 Printing Subsystem, code combinations of the following parameters and statements, in addition to the parameters used for printing on any printer.

```
In any system:
     //ddname  DD    SYSOUT=class,BURST=value,CHARS=table-name,
     //          COPIES=(,(group-value)),FLASH=overlay-name,
     //          MODIFY=(module-name,trc),DCB=OPTCD=J

     //name    OUTPUT  BURST=value,CHARS=table-name,
     //          COPIES=(,(group-value)),FLASH=overlay-name,
     //          MODIFY=(module-name,trc),TRC=value

In a JES2 system:
     /*JOBPARM  BURST=value
```

Most of the formatting parameters can be coded on several statements. If coded more than once for a sysout data set, JES selects one parameter according to override rules and uses it.

The BURST parameter coded on the /*JOBPARM statement applies to all sysout data sets printed on 3800 printers in the job.

Copy Modification

For sysout data sets printed on a 3800, you can modify selected copies of output by specifying a copy modification module name in the MODIFY parameter. Copy modification allows printing predefined data on all pages of a copy or copies of the data set.

For example, you may want to vary column headings or explanatory remarks on different copies of the same printed page. Or, you may want to personalize copies with the recipient's name, address, and other information. Or, you may want to print blanks or certain characters, such as asterisks, to suppress the printing of variable data on particular copies of a page.

The predefined data is created as a copy modification module and stored in SYS1.IMAGELIB using the IEBIMAGE utility program. For information on using IEBIMAGE, see *DFSMS/MVS Utilities*.

Copy modification is done with other printers by using short or spot carbons in the forms set.

Character Arrangements

Specify in the CHARS parameter character-arrangement tables to be used when printing on a 3800.

For the names of tables for the 3800, see the *3800 Programmer's Guide*. The installation should maintain a list of the names of available tables.

Modifying Character-Arrangement Tables

Using the IEBIMAGE utility program, the installation can modify or construct character-arrangement tables and graphic character modification modules to substitute characters or use installation-designed characters.

Dynamically Selecting Character-Arrangement Tables

To select a character-arrangement table for each logical record in the sysout data set, the second character of each logical record must contain a trc character and you must code either of the following:

- TRC in the OUTPUT JCL statement
- OPTCD=J in the DD statement DCB parameter

For details on using the OPTCD subparameter, see the *3800 Programmer's Guide*.

When Data Set Printed on 3800 or Other Printers

You can code a UCS parameter even though a CHARS parameters is also coded; do this if the output might be printed on a 3800 or some other printer. If a printer

other than the 3800 is used, the system uses the UCS parameter and ignores the CHARS parameter.

If UCS is coded and CHARS is not, and the sysout data set is printed on a 3800, the system uses the UCS value as the default value for the missing CHARS parameter.

Example 1

```
//DD8   DD  SYSOUT=B,BURST=YES,CHARS=(GS10,GU12),
//          COPIES=(,(5)),FLASH=BILL,MODIFY=(IMG9,1)

//OT4   OUTPUT BURST=YES,CHARS=(GS10,GU12),COPIES=(,(5)),
//          FLASH=BILL,MODIFY=(IMG9,1)
```

Use these parameters in any system.

Example 2

```
/*JOBPARM BURST=Y
```

Use this statement only in a JES2 system.

Output Formatting to 3211 Printer with Indexing Feature in a JES2 System

To request that output printed by JES2 on a 3211 Printer with the indexing feature be shifted from the normal page margins, code:

```
To indent left margin:
     //name  OUTPUT INDEX=number

To move right margin:
     //name  OUTPUT LINDEX=number
```

JES2 ignores these parameters if the output is printed on a device other than a 3211. To send a sysout data set to a 3211, specify the output class set aside by the installation for printing on a 3211.

Example 1

```
//OT10  OUTPUT INDEX=6
//DD3   DD     CLASS=W,OUTPUT=*.OT10
```

This example indents the left margin 5 spaces.

Example 2

```
//OT11  OUTPUT LINDEX=9
//DD3   DD     CLASS=W,OUTPUT=*.OT11
```

This example moves the right margin in 8 spaces from the usual location.

Output Formatting to Punch

To format punched output from sysout data sets, code:

```
//ddname DD  SYSOUT=(class,form-name),COPIES=number,
//           FCB=fcb-name,DCB=FUNC=I

//name   OUTPUT COPIES=number,FCB=fcb-name,FORMS=form-name
```

Interpretation of Punched Cards

Cards punched by a 3525 Card Punch are interpreted if JES processes the sysout data set and if the following is coded:

```
//ddname  DD SYSOUT=class,DCB=FUNC=I
```

If the data set is punched on a different card punch, JES ignores the FUNC=I subparameter.

The installation can define a special output class for 3525 output.

Card interpretation by an external writer is an operator-specified function.

Interpretation in a JES3 System

Punched output may or may not be interpreted depending on the installation-defined standard for the output class.

Examples

```
//DD17   DD  SYSOUT=(Q,PUN6),COPIES=5,
//        FCB=IMG4,DCB=FUNC=I

//OT3    OUTPUT COPIES=5,FCB=IMG4,FORMS=PUN6
//DD18   DD     SYSOUT=Q,OUTPUT=*.OT3,DCB=FUNC=I
```

Output Formatting of Dumps on 3800 Printing Subsystem

You can request a high-density dump on the 3800 through two parameters on the DD statement for the dump data set or on an OUTPUT JCL statement referenced by the dump DD statement:

- FCB=STD3. This parameter produces dump output at 8 lines per inch.
- CHARS=DUMP. This parameter produces 204-character print lines.

You can code one or both of these parameters. You can place both on the same statement or one on each statement.

Examples

```
//SYSABEND DD SYSOUT=J,FCB=STD3,CHARS=DUMP

//DUMPOT OUTPUT FCB=STD3,CHARS=DUMP
//SYSABEND DD  SYSOUT=J,OUTPUT=*.DUMPOT
```

Chapter 26. Sysout Resources - Output Limiting

Figure 26-1. Output Limiting Task for Requesting Sysout Data Set Resources

TASKS FOR REQUESTING SYSOUT RESOURCES	STATEMENTS AND PARAMETERS FOR TASK				
	JCL Statements			JES2 Statements	JES3 Statements
	DD	OUTPUT JCL	Other JCL		
Output Limiting					
	OUTLIM		lines and cards on JOB JES2 accounting information BYTES, CARDS, LINES, and PAGES on JOB	BYTES, CARDS, LINES, and PAGES on /*JOBPARM	BYTES, CARDS, LINES, and PAGES on //*MAIN

Output Limiting

To limit the number of logical records in a sysout data set, specify a maximum number of records to be written to a sysout data set or to all sysout data sets in a job.

By establishing a limit, you avoid printing a useless, huge listing if your program enters an endless loop that contains a write instruction to a sysout data set. After reaching the limit, the system abnormally terminates the step, or sends a warning message to the operator.

Limiting Output in an APPC Scheduling Environment

To limit the output for a job in an APPC scheduling environment, use the DD statement OUTLIM parameter or the JOB statement BYTES, CARDS, LINES, and PAGES parameters.

The DD OUTLIM parameter limits the number of logical records in a single sysout data set, or in an internal reader data set. Code the DD statement as follows:

```
//ddname  DD  SYSOUT=class,OUTLIM=number
```

Use the JOB statement BYTES, CARDS, LINES, or PAGES parameter to limit the number of logical records written to all sysout data sets in a job. Code the job statement as follows:

```
//JOB1   JOB  accounting-info,programmer,BYTES=(number)

//JOB2   JOB  accounting-info,programmer,CARDS=(number)

//JOB3   JOB  accounting-info,programmer,LINES=(number)

//JOB4   JOB  accounting-info,programmer,PAGES=(number)
```

In an APPC scheduling environment, you cannot use JES control statements to limit output. If you code a JES2 control statement in an APPC scheduling

© Copyright IBM Corp. 1988, 1999

environment, it will cause a JCL error. If you code a JES3 control statement, the system will ignore it and the statement will appear as a comment in the job listing.

Limiting Output in a Non-APPC Scheduling Environment

Valid parameters in a non-APPC scheduling environment include the DD OUTLIM parameter and the JOB statement BYTES, CARDS, LINES, and PAGES parameters described in “Limiting Output in an APPC Scheduling Environment” on page 26-1. In addition, you can code JES control statements to limit the output for your job.

```
For all sysout data sets in a job in a JES2 system:
     //jobname  JOB (,,,lines,cards),...
     /*JOBPARM  BYTES=number
     /*JOBPARM  CARDS=number
     /*JOBPARM  LINES=number
     /*JOBPARM  PAGES=number

For all sysout data sets in a job in a JES3 system:
     //*MAIN  BYTES=number
     //*MAIN  CARDS=number
     //*MAIN  LINES=number
     //*MAIN  PAGES=number
```

The system limits output based on the limit specified on the JOB statement. If you do not code a JOB statement limit, the system uses the limit specified on the //*MAIN or /*JOBPARM statements. If you do not code a limit on the JOB, /*JOBPARM, or //*MAIN statements, the system uses the installation default limit, specified at JES initialization.

Actions when Limit Exceeded

On the JOB statement parameters and the JES3 //*MAIN statement, you can indicate the action that the system is to take when the output limit is exceeded.

WARNING: The system issues a warning message to the operator.
CANCEL: The system terminates the job.
DUMP: The system terminates the job and dumps the step being executed when the limit was exceeded.

Example 1

```
//DD1   DD   SYSOUT=T,OUTLIM=3000
```

Use this example in any system.

Example 2

```
//JOBA  JOB  (,,,4,2000),'T. KATZ'

/*JOBPARM  BYTES=40

/*JOBPARM  CARDS=2000

/*JOBPARM  LINES=4

/*JOBPARM  PAGE=400
```

Use these examples in a JES2 system.

Example 3

```
//*MAIN  BYTES=(40,WARNING)

//*MAIN  CARDS=(20,CANCEL)

//*MAIN  LINES=(4,DUMP)

//*MAIN  PAGES=(400,WARNING)
```

Use these examples in a JES3 system.

Example 4

```
//JOB1  JOB  BYTES=(40,WARNING)

//JOB2  JOB  CARDS=(20,CANCEL)

//JOB3  JOB  LINES=(4,DUMP)

//JOB4  JOB  PAGES=(400,WARNING)
```

Use these examples in any system.

Chapter 27. Sysout Resources - USERDATA OUTPUT JCL Keyword

The information provided on the user-oriented keyword, USERDATA, is defined and used by the installation. The installation can use certain JES or PSF installations exits to access the keyword specification.

References

Refer to the following manuals for additional information on potential uses for the USERDATA keyword.

- *OS/390 JES2 Installation Exits* and *OS/390 JES2 Macros*, section “Choosing Which Exits to Implement” lists JES2 installation exits 1, 15 and 23 as pertaining to SYSOUT separator page processing.
- *OS/390 JES3 Customization*, section “Installation Exits Listed by JES3 Function” lists JES3 installation exits 20, 21, 23 and 45 as pertaining to SYSOUT separator page processing.
- *PSF/MVS System Programming Guide*, lists PSF installation exits 1, 2 and 3 as pertaining to SYSOUT separator page processing.

Examples

Example 1

```
//OUTUSER1  OUTPUT USERDATA='My Own Installation Sub-Title',
//            TITLE='My Own SYSOUT Title'
//DD1       DD     SYSOUT=A,OUTPUT=*.OUTUSER1
```

In this example, the SYSOUT data set DD1 refers to the OUTPUT JCL statement named OUTUSER1. If the installation intended to print the USERDATA value on the SYSOUT data set separator page, and if the installation coded the necessary changes to the JES and PSF SYSOUT data set separator page exits, the TITLE value enclosed within the apostrophes (My Own SYSOUT Title) would be printed on the SYSOUT data set separator page. In addition, the USERDATA value enclosed within the apostrophes (My Own Installation Sub-Title) would be printed on the SYSOUT data set separator page.

Example 2

```
//OUTUSER2  OUTPUT USERDATA='LOCALDEV=Option1'
//DD2       DD     SYSOUT=A,OUTPUT=*.OUTUSER2
```

In this example, the SYSOUT data set DD2 refers to the OUTPUT JCL statement named OUTUSER2. If the installation defined its own keyword (LOCALDEV) and the valid values for the keyword, and if the installation made the necessary changes to the appropriate JES and PSF exits, the installation would have to parse the USERDATA value to determine if the installation keyword and value were specified. The LOCALDEV keyword value of Option1 could then be used by the installation.

© Copyright IBM Corp. 1988, 1999

Part 6. Examples

Chapter 28. Example - Assemble, Linkedit, and Go 28-1

Chapter 29. Example - Multiple Output 29-1

Chapter 30. Example - Obtaining Output in a JES2 System 30-1

Chapter 31. Example - Obtaining Output in a JES3 System 31-1

Chapter 32. Example - Identifying Data Sets to the System 32-1

This part contains examples of sets of job control statements. Some are for useful processing, some show particular techniques. For examples of the job control statements needed to use utilities, see *DFSMS/MVS Utilities*.

© Copyright IBM Corp. 1988, 1999

Part 6. Examples

Chapter 28. Example - Assemble, Linkedit, and Go

Example 1

The following example uses the COND parameter to conditionally execute job steps.

```
//USUAL     JOB   A2317P,'MAE BIRDSALL'
//ASM       EXEC  PGM=IEV90,REGION=256K,          EXECUTES ASSEMBLER
//            PARM=(OBJECT,NODECK,'LINECOUNT=50')
//SYSPRINT  DD    SYSOUT=*,DCB=BLKSIZE=3509  PRINT THE ASSEMBLY LISTING
//SYSPUNCH  DD    SYSOUT=B                   PUNCH THE ASSEMBLY LISTING
//SYSLIB    DD    DSNAME=SYS1.MACLIB,DISP=SHR    THE MACRO LIBRARY
//SYSUT1    DD    DSNAME=&&SYSUT1,UNIT=SYSDA,    A WORK DATA SET
//            SPACE=(CYL,(10,1))
//SYSLIN    DD    DSNAME=&&OBJECT,UNIT=SYSDA,  THE OUTPUT OBJECT MODULE
//            SPACE=(TRK,(10,2)),DCB=BLKSIZE=3120,DISP=(,PASS)
//SYSIN     DD    *                              IN-STREAM SOURCE CODE
            .
            .
            code
            .
/*
//LKED      EXEC  PGM=HEWL,                     EXECUTES LINKAGE EDITOR
//            PARM='XREF,LIST,LET',COND=(8,LE,ASM)
//SYSPRINT  DD    SYSOUT=*                       LINKEDIT MAP PRINTOUT
//SYSLIN    DD    DSNAME=&&OBJECT,DISP=(OLD,DELETE) INPUT OBJECT MODULE
//SYSUT1    DD    DSNAME=&&SYSUT1,UNIT=SYSDA,    A WORK DATA SET
//            SPACE=(CYL,(10,1))
//SYSLMOD   DD    DSNAME=&&LOADMOD,UNIT=SYSDA,   THE OUTPUT LOAD MODULE
//            DISP=(MOD,PASS),SPACE=(1024,(50,20,1))
//GO        EXEC  PGM=*.LKED.SYSLMOD,TIME=(,30), EXECUTES THE PROGRAM
//            COND=((8,LE,ASM),(8,LE,LKED))
//SYSUDUMP  DD    SYSOUT=*                       IF FAILS, DUMP LISTING
//SYSPRINT  DD    SYSOUT=*,                      OUTPUT LISTING
//            DCB=(RECFM=FBA,LRECL=121)
//OUTPUT    DD    SYSOUT=A,                      PROGRAM DATA OUTPUT
//            DCB=(LRECL=100,BLKSIZE=3000,RECFM=FBA)
//INPUT     DD    *                              PROGRAM DATA INPUT
            .
            .
            data
            .
/*
//
```

This example shows JCL that can be used to:

- Assemble object code entered in the input stream: the step named ASM.
- Link edit the object module, if the assembly did not result in a return code of 8 or higher: the step named LKED.
- Execute the link edited module, if neither the assembly nor the linkage editing resulted in a return code of 8 or higher: the step named GO.

Example 2

© Copyright IBM Corp. 1988, 1999

Example - Assemble, Linkedit, and Go

The following example of Assemble, Linkedit, and Go uses the IF/THEN/ELSE/ENDIF statement construct to conditionally execute job steps.

```
//USUAL     JOB   A2317P,'MAE BIRDSALL'
//ASM       EXEC  PGM=IEV90,REGION=256K,         EXECUTES ASSEMBLER
//           PARM=(OBJECT,NODECK,'LINECOUNT=50')
//SYSPRINT  DD    SYSOUT=*,DCB=BLKSIZE=3509  PRINT THE ASSEMBLY LISTING
//SYSPUNCH  DD    SYSOUT=B                   PUNCH THE ASSEMBLY LISTING
//SYSLIB    DD    DSNAME=SYS1.MACLIB,DISP=SHR    THE MACRO LIBRARY
//SYSUT1    DD    DSNAME=&&SYSUT1,UNIT=SYSDA,    A WORK DATA SET
//           SPACE=(CYL,(10,1))
//SYSLIN    DD    DSNAME=&&OBJECT,UNIT=SYSDA,  THE OUTPUT OBJECT MODULE
//           SPACE=(TRK,(10,2)),DCB=BLKSIZE=3120,DISP=(,PASS)
//SYSIN     DD    *                              IN-STREAM SOURCE CODE
            .
            .
            code
            .
/*
//RC1OK     IF (ASM.RC LT 8) THEN            EVALUATES RC FROM STEP ASM
//LKED      EXEC  PGM=HEWL,                     EXECUTES LINKAGE EDITOR
//           PARM='XREF,LIST,LET'
//SYSPRINT  DD    SYSOUT=*                       LINKEDIT MAP PRINTOUT
//SYSLIN    DD    DSNAME=&&OBJECT,DISP=(OLD,DELETE) INPUT OBJECT MODULE
//SYSUT1    DD    DSNAME=&&SYSUT1,UNIT=SYSDA,    A WORK DATA SET
//           SPACE=(CYL,(10,1))
//SYSLMOD   DD    DSNAME=&&LOADMOD,UNIT=SYSDA,   THE OUTPUT LOAD MODULE
//           DISP=(MOD,PASS),SPACE=(1024,(50,20,1))
//RC2OK         IF (LKED.RC LT 8) THEN
//GO             EXEC  PGM=*.LKED.SYSLMOD,TIME=(,30), EXECUTES PROGRAM
//SYSUDUMP       DD    SYSOUT=*                 IF FAILS, DUMP LISTING
//SYSPRINT       DD    SYSOUT=*,                OUTPUT LISTING
//           DCB=(RECFM=FBA,LRECL=121)
//OUTPUT         DD    SYSOUT=A,                PROGRAM DATA OUTPUT
//           DCB=(LRECL=100,BLKSIZE=3000,RECFM=FBA)
//INPUT          DD    *                        PROGRAM DATA INPUT
                 .
                 .
                 data
                 .
/*
//ENDRC2        ENDIF
//ENDRC1    ENDIF
//
```

This example shows JCL that can be used to:

- Assemble object code entered in the input stream: the step named ASM.
- Link edit the object module, if the assembly resulted in a return code of lower than 8: the step named LKED.
- Nest IF/THEN/ELSE/ENDIF statement constructs
- Execute the link edited module, if the assembly and the linkage editing resulted in a return code of lower than 8: the step named GO.

Chapter 29. Example - Multiple Output

```
//EXAMP    JOB     MSGCLASS=A
//OUT1     OUTPUT  DEFAULT=YES,DEST=COMPLEX7,FORMS=BILLING,
//           CHARS=(AOA,AOB),COPIES=2
//OUT2     OUTPUT  DEFAULT=YES,DEST=COMPLEX3
//OUT3     OUTPUT  DEST=COMPLEX1
//STEP1    EXEC    PGM=ORDERS
//OUT4     OUTPUT  DEFAULT=YES,DEST=COMPLEX9
//R1       DD      SYSOUT=A,OUTPUT=*.OUT3
//R2       DD      SYSOUT=A
//STEP2    EXEC    PGM=BILLING
//B1       DD      SYSOUT=A
//B2       DD      SYSOUT=A
```

This job requests that the system produce nine sets of output: eight sets of job output and one set for the system-managed output data set.

Set 1

In STEP1, DD statement R1 explicitly references OUTPUT JCL statement OUT3. Therefore, the system produces one set of output at COMPLEX1 for DD statement R1 combined with OUTPUT JCL statement OUT3.

Set 2

In STEP1, DD statement R2 implicitly references OUTPUT JCL statement OUT4 for both of the following reasons:

- DD statement R2 does not contain an OUTPUT parameter.
- STEP1 contains an OUTPUT JCL statement with DEFAULT=YES.

Therefore, the system produces one set of output at COMPLEX9 for DD statement R2 combined with OUTPUT JCL statement OUT4.

Sets 3 through 8

In STEP2, DD statements B1 and B2 implicitly reference OUTPUT JCL statements OUT1 and OUT2 for all of the following reasons:

- DD statements B1 and B2 do not contain OUTPUT parameters.
- STEP2 does not contain an OUTPUT JCL statement with DEFAULT=YES.
- DEFAULT=YES is specified on OUTPUT JCL statements OUT1 and OUT2.

Therefore, the system produces three sets of output each for DD statements B1 and B2:

Sets 3 and 4 at COMPLEX7 for DD statement B1 combined with OUTPUT JCL statement OUT1.
Set 5 at COMPLEX3 for DD statement B1 combined with OUTPUT JCL statement OUT2.
Sets 6 and 7 at COMPLEX7 for DD statement B2 combined with OUTPUT JCL statement OUT1.
Set 8 at COMPLEX3 for DD statement B2 combined with OUTPUT JCL statement OUT2.

Set 9

The system-managed output data set is processed locally because of the MSGCLASS parameter on the JOB statement.

© Copyright IBM Corp. 1988, 1999

Example - Multiple Output

Chapter 30. Example - Obtaining Output in a JES2 System

```
/*PRIORITY 5
//OUTJOB  JOB  BAKER,PERFORM=100,MSGCLASS=J
/*SETUP   SCHLIB
/*JOBPARM COPIES=2,LINECT=20,ROOM=223,FORMS=GRN1
//OUT1    OUTPUT JESDS=ALL
//OUT2    OUTPUT DEST=PRINTER8,FCB=STD3,FORMS=2PRT,UCS=TN
//STEP1   EXEC  PGM=TESTSYSO
//DD1    DD DSN=DATA,DISP=OLD,UNIT=3350,VOL=SER=SCHLIB
//DD2     DD    DSN=&&TEMP,UNIT=3350,DISP=(NEW,DELETE),
        SPACE=(TRK,(10,5))
//DD3     DD    SYSOUT=A,OUTPUT=*.OUT2
//DD4     DD    SYSOUT=(A,,GRPH)
//DD5     DD    SYSOUT=L,OUTPUT=*.OUT1,DEST=HDQ
```

This example shows the use of JES2 and JCL statements to obtain output.

1. The job will be selected at priority level 5.
2. The job will run in performance group 100; the meaning of 100 is defined by the installation. All system messages are to be written to output class J.
3. The JOBPARM statement indicates that:
   a. Two copies of the entire job-related output will be printed.
   b. No more than 20 lines per page will be printed (LINECT=20). You can override this LINECT parameter by coding the LINECT parameter on the OUTPUT JCL statement.
   c. The programmer's room number is 233. This appears on the separator page and is used for distributing output.
   d. Forms name GRN1 is the name of the form to be used by all data sets unless a specific form is defined on a DD, JES2 /*OUTPUT, or JCL OUTPUT statement.
4. The OUTPUT JCL statement OUT2 indicates that:
   a. The destination for the output is PRINTER8. PRINTER8 does not necessarily have to be defined as a printer, it can be defined as any output device.
   b. If the printer has the forms control buffer feature, STD3 must be the name of a member of SYS1.IMAGELIB. STD3 defines the special forms control buffer image to be used for processing any data set that has *.OUT2 coded in the SYSOUT parameter.
   c. Forms name 2PRT is the name of the form JES2 uses for printing any data sets that have *.OUT2 coded in the SYSOUT parameter (for example, DD3).
   d. TN is the train or UCS used in output processing.
5. The SETUP statement indicates that volume SCHLIB should be mounted before this job begins processing.
6. SYSOUT data sets (except DD3 and DD4) are printed on the form called GRN1. The DD4 SYSOUT data set is printed on the form called GRPH; the

© Copyright IBM Corp. 1988, 1999

DD3 SYSOUT data set is printed on the form called 2PRT because the code name subparameter of DD3 contains the value *.OUT2 (referring to the OUTPUT JCL statement).

7. The output data set from DD5 and the accompanying data sets will be sent to HDQ.

Chapter 31. Example - Obtaining Output in a JES3 System

```
//OUTJOB  JOB  BAKER,PERFORM=100,MSGCLASS=J
//*FORMAT PR,DDNAME=,COPIES=2,FORMS=GRN1
//*FORMAT PR,DDNAME=DD3,DEST=PRINTER8,CARRIAGE=STD3,
//*FORMS=2PRT,TRAIN=TN
//STEP1   EXEC PGM=TESTSYSO
//DD1     DD   DSN=DATA,UNIT=3350,VOL=SER=SCHLIB,
//           DISP=(OLD,KEEP),SPACE=(TRK,(5,2))
//DD2     DD   DSN=&TEMP,UNIT=3350,DISP=(NEW,DELETE),
//           SPACE=(TRK,(10,5))
//DD3     DD   SYSOUT=(A)
//DD4     DD   SYSOUT=(A,,GRPH)
//DD5     DD   SYSOUT=L
```

This example shows some of the JES3 and JCL statements that can be used to obtain output.

1. All system messages are to be written to output class J.
2. The first //*FORMAT statement indicates that:
   a. All print data sets (according to class) that do not have //*FORMAT statements will be printed according to the parameters on this statement unless the output class defines specific processing characteristics because DDNAME is coded without a name (DDNAME=,) and applies to all output data sets for the job.
   b. JES3 uses the form named GRN1 and prints two copies of all data sets unless a specific form or number of copies is defined on a DD statement or for a class by the installation.
3. The second //*FORMAT statement indicates that:
   a. The destination for the output is a printer that has an installation-defined name of PRINTER8.
   b. If PRINTER8 has the forms control buffer feature, STD3 must be the name of a member of SYS1.IMAGELIB. STD3 defines the special forms control buffer image or carriage tape to be used for processing the job.
   c. Forms name 2PRT is the name of the forms for DD3.
   d. TN means test printing on a 1403, 3211, or 3203-5 printer.

© Copyright IBM Corp. 1988, 1999

Chapter 32. Example - Identifying Data Sets to the System

```
/*PRIORITY    8
//DATASETS  JOB  FREEMAN,MSGLEVEL=1
//STEP1     EXEC PGM=IEFBR14
//D1        DD   DSN=ABC,DISP=(NEW,CATLG),UNIT=3350,
//            VOL=SER=333001,SPACE=(CYL,(12,1,1),CONTIG)
//D2        DD   DSN=&&NAME,UNIT=3330,SPACE=(TRK,(10,1))
//D3        DD   DSN=SYSLIB,DISP=(OLD,KEEP)
//D4        DD   *
            .
            .
            .
          data
            .
            .
            .
/*
```

1. This job runs in priority 8, the meaning of which is defined by the installation.
2. The job statement specifies that system messages and JCL statements are to be printed (MSGLEVEL=1).
3. D1 catalogs a newly created data set. The space request is for 12 primary cylinders, 1 secondary, 1 directory, and the space is to be contiguous.
4. D2 creates a temporary data set on a 3330. The space request is for 10 primary tracks and 1 secondary.
5. D3 defines an old cataloged data set.
6. D4 defines a SYSIN data set. This will be followed by data in the input stream.

© Copyright IBM Corp. 1988, 1999

Appendixes

© Copyright IBM Corp. 1988, 1999

Appendix A. Indexed Sequential Data Sets

Note that SMS does not manage ISAM data sets.

Indexed sequential (ISAM) data sets are created and retrieved using special subsets of DD statement parameters and subparameters. Each data set can occupy up to three different areas:

- **Index area:** This area contains master and cylinder indexes associated with the data set. It exists for any indexed sequential data set that has a prime area occupying more than one cylinder.
- **Prime area:** This area contains data and related track indexes. It exists for all indexed sequential data sets.
- **Overflow area:** This area contains overflow from the prime area when new data is added. It is optional.

Volumes for ISAM Data Sets

Indexed sequential data sets must reside on direct access volumes. The data set can reside on more than one volume and the volumes may, in some cases, be on different types of devices. If the volumes have indexed volume tables of contents (VTOCs), the ISAM index area must reside on the first volume.

Creating an Indexed Sequential Data Set

One to three DD statements are used to define a new indexed sequential data set; each statement defines a different area.

Three DD statements
Define the areas in the following order:

1. Index area
2. Prime area
3. Overflow area

Two DD statements
Define the areas in the following order:

1. Index area
2. Prime area

Or

1. Prime area and, optionally, index area
2. Overflow area

One DD statement
The statement defines the prime area and, optionally, the index area.

When more than one DD statement is used to define the data set, assign a ddname only to the first DD statement; the name field of the other statements must be blank.

The only DD statement parameters that can be coded when defining a new indexed sequential data set are:

© Copyright IBM Corp. 1988, 1999

AVGREC	LABEL	RETPD
DCB	LIKE	SECMODEL
DISP	LRECL	SPACE
DSNAME	RECFM	UNIT
EXPDT	REFDD	VOLUME

DSNAME Parameter

The DSNAME parameter is required on any DD statement that defines a new temporary or permanent indexed sequential data set. Code:

```
//ddname DD  DSNAME=name(INDEX)
//       DD  DSNAME=name(PRIME)
//       DD  DSNAME=name(OVFLOW)
```

If you are using only one DD statement, code either:

```
//ddname DD  DSNAME=name(PRIME)

//ddname DD  DSNAME=name
```

When you reuse previously allocated space to create an indexed sequential data set, the DSNAME parameter must contain the name of the old data set to be overlaid.

UNIT Parameter

The UNIT parameter is required on any DD statement that defines a new indexed sequential data set, unless VOLUME=REF=reference is coded. You must request a direct access device in the UNIT parameter. Do not code DEFER.

If the prime and index areas are defined on separate DD statements, request the same number of direct access devices for the prime area as volumes specified in the VOLUME parameter. Request only one direct access volume for an index area and one for an overflow area.

A DD statements for the index area or overflow area can request a device type different than the type requested on the other statements.

VOLUME Parameter

The VOLUME parameter is required if you want an area of the data set written on a specific volume or the prime area requires the use of more than one volume. If the prime area and index area are defined on the same statement, you cannot request more than one volume on the DD statement. Either supply the volume serial number(s) in the VOLUME parameter or code VOLUME=REF=reference. In all cases, you can specify PRIVATE in the VOLUME parameter.

Note:

- If a nonspecific volume request is used when creating a new indexed sequential data set and its DSNAME already exists on a volume eligible for allocation, the job will fail if the system places the new data set on that volume. However, if the old data set with the duplicate name is on a volume other than the one selected for the new data set, the new data set is not affected and will be added to the volume. You can correct job failures caused by duplicate names by scratching the old data set or by renaming the new data set, then resubmitting the job.

- The system fails to allocate space for a new indexed sequential data set with a nonspecific volume request when none of the volumes eligible for allocation contain enough space.
- If the first volume selected by allocation to satisfy a request for a new indexed sequential data set does not contain enough space to satisfy the request, the system does not try to find another volume with enough space if either of these conditions is met:
  - The request is for multiple volumes or units
  - The request uses more than one DD statement to define the data set.

LABEL Parameter

The LABEL parameter is needed only to specify a retention period, EXPDT or RETPD, or password protection, PASSWORD.

DCB Parameter

You must code the DCB parameter on every DD statement that defines an indexed sequential data set. At minimum, the DCB parameter must contain DSORG=IS or DSORG=ISU. Other DCB subparameters can be coded to complete the data control block, if the processing program does not complete it.

When more than one DD statement is used to define the data set, code all the DCB subparameters on the first DD statement. On the other DD statements, refer to the DCB parameter on the first statement by coding:

```
DCB=*.ddname
```

When reusing previously allocated space and recreating an indexed sequential data set, desired changes in the DCB parameter must be coded on the DD statement. Although you are creating a new data set, some DCB subparameters cannot be changed if you want to use the space the old data set used. The DCB subparameters you can change are:

BFALN	DSORG	NCP	RECFM
BLKSIZE	KEYLEN	NTM	RKP
CYLOFL	LRECL	OPTCD	

DISP Parameter

If you are creating a new data set and not reusing preallocated space, the DISP parameter is needed only if you want to:

Keep the data set	DISP=(,KEEP)
Catalog the data set	DISP=(,CATLG)
Pass the data set	DISP=(,PASS)

If you are reusing previously allocated space and recreating an indexed sequential data set, code DISP=OLD. The newly created data set will overlay the old one.

In order to catalog the data set by coding DISP=(,CATLG) or to pass the data set by coding DISP=(,PASS), you must define the data set on only one DD statement. If you define the data set on more than one DD statement and the volumes containing the data set are on the same device type, use the access method services DEFINE command to catalog the data set. For details, refer to

DFSMS/MVS Access Method Services for ICF,
DFSMS/MVS Access Method Services for VSAM.

SPACE Parameter

The SPACE parameter is required on any DD statement that defines a new indexed sequential data set. Either ask the system to assign the space or request specific tracks. If you use more than one DD statement to define the data set, each DD statement must request space in the same way.

System Assignment of Space

You must request the primary quantity in cylinders, CYL. When the DD statement that defines the prime area requests more than one volume, each volume is assigned the number of cylinders requested in the SPACE parameter.

The *index* subparameter is used to indicate how many cylinders are required for an index. When you use one DD statement to define the prime and index areas and you want to explicitly state the size of the index, code the index subparameter.

You can code the CONTIG subparameter in the SPACE parameter. However, if you code CONTIG on one of the statements, you must code it on all of them.

You cannot request a secondary quantity for an indexed sequential data set. Also, you cannot code the subparameters RLSE, MXIG, ALX, and ROUND.

Specific Track Request

The number of tracks requested must be equal to one or more whole cylinders. The address of the beginning track must be the first track of a cylinder other than the first cylinder on the volume. When the DD statement that defines the prime area requests more than one volume, space is allocated for the prime area beginning at the specified address and continuing through the volume and onto the next volume until the request is satisfied. This can be done only if the volume table of contents of the second and all succeeding volumes is contained in the first cylinder of each volume.

Use the index subparameter to indicate how many tracks the index requires. The number of tracks specified must be equal to one or more cylinders. When you use one DD statement to define the prime and index areas and you want to state the size of the index, code the index subparameter.

Procedure when Allocation Error Occurs

If a new indexed sequential data set is to reside on more than one volume and an error occurs during volume allocation, do the following before resubmitting the job: Use the IEHPROGM utility program to scratch the data set labels on any of the volumes to which the data set was successfully allocated. This utility program is described in *DFSMS/MVS Utilities*.

Area Arrangement of an Indexed Sequential Data Set

When creating an indexed sequential data set, the arrangement of the areas is based on:

- The number of DD statements used to define the data set
- What area each DD statement defines

The system uses an additional criterion when the index area is not defined on a separate DD statement: Is an index size coded in the SPACE parameter on the DD statement that defines the prime area?

Figure A-1 illustrates the different arrangements that can result based on these criteria. In addition, it indicates what restrictions apply on the number and types of devices that can be requested.

Figure A-1. Area Arrangement of ISAM Data Sets

Criteria			Restrictions on	Resulting
Number of DD statements	**Area defined on DD statement**	**Index size coded?**	**Device Types and Number of Devices Requested**	**Arrangement of Areas**
3	INDEX PRIME OVFLOW	-	None	Separate index, prime, and overflow areas.
2	INDEX PRIME	-	None	Separate index and prime areas.[1]
2	PRIME OVFLOW	No	None	Separate prime and overflow areas. An index area is at the end of the overflow area.
2	PRIME OVFLOW	Yes	The statement for the prime area cannot request more than one device.	Separate prime and overflow areas. An index area is embedded in the prime area.
1	PRIME	No	None	Prime area with index area at its end.[2]
1	PRIME	Yes	The statement cannot request more than one device.	Prime area with embedded index area.[2]

[1] If both areas are on volumes on the same device type and if one of the cylinders allocated for the index area is only partially filled, the system establishes the overflow area in the unused portion of that cylinder.

[2] If the index area occupies at least one cylinder and if the unused portion of the index area is less than one cylinder, the unused portion is established as an overflow area. For a one-cylinder data set, no overflow area is established.

Retrieving an Indexed Sequential Data Set

If all areas of an existing indexed sequential data set are on volumes of the same device type, you can retrieve the entire data set with one DD statement. If the index or overflow is on a volume of a different device type, use two DD statements. If the index and overflow are on volumes of different device types, use three DD statements to retrieve the data set. The DD statements are coded in the following order:

1. Index area
2. Prime area
3. Overflow area

The **only** DD statement parameters that you may code when retrieving an indexed sequential data set are:

DSNAME
UNIT
VOLUME
DCB
DISP

DSNAME Parameter

The DSNAME parameter is always required. Identify the data set by its name. Do not code INDEX, PRIME, or OVFLOW. If the data set was passed from a previous step, identify it by a backward reference.

UNIT Parameter

The UNIT parameter must be coded, unless the data set resides on one volume and was passed. Specify in the UNIT parameter the device type and the unit-count, if more than one device is required.

If the data set is on more than one volume but the volumes are for the same device type, you need only one DD statement to retrieve the data set. Request one device per volume in the UNIT parameter.

If the areas are on different types of devices, code a DD statement for each different device type.

Another way to request a device is to code UNIT=AFF=ddname, where the referenced DD statement requests direct access.

VOLUME Parameter

The VOLUME parameter must be coded, unless the data set is on one volume and was passed from a previous step. Identify in the VOLUME parameter the serial numbers of the volumes on which the data set resides. Code the serial numbers in the same order that they were coded on the DD statements used to create the data set.

DCB Parameter

The DCB parameter must always contain DSORG=IS or DSORG=ISU. Do not code other DCB subparameters if the data set is passed from a previous step or is cataloged. However, you can code other DCB subparameters to complete the data control block, if it is not completed in the processing program.

DISP Parameter

The DISP parameter must always be coded. The first subparameter of the DISP parameter must be SHR or OLD.

When you are updating an existing indexed sequential data set, code DISP=OLD. If you specify DISP=SHR, the data set will not open correctly.

Optionally, you can specify a disposition in the second subparameter.

Figure A-2. DD Parameters for Retrieving or Extending an ISAM Data Set

Area	Parameter	Comments
INDEX (coded only if index area is not on same device type as prime area)	DSNAME	Required. Code the same name as in the second DD statement.
	DISP	Required. Code the same value as in the second DD statement.
First DD statement	UNIT	Required
	VOLUME	Required
	DCB	Required
PRIME; or	DSNAME	Required
PRIME with overflow; or	DISP	Required. Specifies whether data set is being retrieved or updated.
PRIME with overflow and index	UNIT	Required, unless passed data set is being retrieved and all three areas are on one volume.
Second or only DD statement	VOLUME	Same requirement as UNIT. If coded, list volumes in the order in which they were defined.
	DCB	Required
OVFLOW (coded only if overflow area is not on same device type as prime area)	DSNAME	Required. Code the same value as in the second DD statement.
	DISP	Required. Code the same value as in the second DD statement.
Third DD statement	UNIT	Required
	VOLUME	Required
	DCB	Required

Example 1

```
//ISAMJOB  JOB  ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1    EXEC PGM=INCLUDE
//DD1      DD   DSNAME=DATASET1(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//          VOLUME=SER=777777,SPACE=(CYL,(10),,CONTIG),
//          DCB=(DSORG=IS,RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//         DD   DSNAME=DATASET1(PRIME),DISP=(NEW,KEEP),UNIT=3330,
//          VOLUME=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
//         DD   DSNAME=DATASET1(OVFLOW),DISP=(NEW,KEEP),UNIT=3330,
//          VOLUME=REF=*.DD1,SPACE=(CYL,(25),,CONTIG),DCB=*.DD1
```

This example creates an indexed sequential data set on one 3330 volume.

Example 2

```
//RETRISAM JOB  ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1    EXEC PGM=RETRIEVE
//DDISAM   DD   DSNAME=DATASET1,DCB=DSORG=IS,UNIT=3330,DISP=OLD,
//          VOLUME=SER=777777
```

This example job shows the DD statements needed to retrieve the indexed sequential data set created in the first example.

Example 3

```
//ISAMJOB JOB  ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1   EXEC PGM=IEFISAM
//DDISAM  DD   DSNAME=DATASET2(INDEX),DISP=(NEW,KEEP),UNIT=3330,
//         VOLUME=SER=888888,SPACE=(CYL,10,,CONTIG),DCB=(DSORG=IS,
//         RECFM=F,LRECL=80,RKP=1,KEYLEN=8)
//        DD   DSNAME=DATASET2(PRIME),DISP=(,KEEP),UNIT=3350,
//         VOLUME=SER=999999,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
//        DD   DSNAME=DATASET2(OVFLOW),DISP=(,KEEP),UNIT=3350,
//         VOLUME=SER=AAAAAA,SPACE=(CYL,10,,CONTIG),DCB=*.DDISAM
```

This job creates an indexed sequential data set on one 3330 and two 3350 volumes.

Example 4

```
//RERISAM JOB  ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1   EXEC PGM=IEFISAM
//DDISAM  DD   DSNAME=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=3330,
//         VOLUME=SER=888888
//        DD   DSNAME=DATASET2,DCB=DSORG=IS,DISP=OLD,UNIT=(3350,2),
//         VOLUME=SER=(999999,AAAAAA)
```

This job shows the DD statements needed to retrieve the indexed sequential data set created in the previous example.

Example 5

```
//CATISAM  JOB  ,,MSGLEVEL=(1,1),PERFORM=25
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 DEFINE NONVSAM (NAME(DATASET2) DEVT(3330 3350 3350) -
                 VOL(888888 999999 AAAAAA) )
/*
```

This job catalogs a data set previously created on 3330 and 3350 volumes. (See the third example, jobname ISAMJOB.)

Appendix B. Generation Data Sets

This appendix describes both SMS-managed and non-SMS-managed generation data sets.

A generation data set is one of a collection of successive, historically related, cataloged data sets, known as a generation data group (GDG). The system keeps track of each data set in a generation data group as it is created, so that new data sets can be chronologically ordered and old ones easily retrieved.

Note: A VSAM data set cannot be a generation data set.

To create or retrieve a generation data set, follow the generation data group name in the DD statement DSNAME parameter with a relative generation number. When you catalog the generation data set, the operating system uses that number to construct a four-digit absolute generation number and a two-digit version number, resulting in a number of the form G0000V00 to represent that generation. The G0000V00 number must be unique within the GDG so that the system can sort the data sets into the correct chronological sequence unambiguously.

WARNING: IBM strongly recommends that you specify a new generation by a relative generation number. This avoids the possibility of causing the generation number of any data set in the GDG to exceed 9000, which might cause an ambiguity regarding the correct chronological order. This could happen, for example, if you specified a fully-qualified name and used the first two digits of the number to represent the year. If, however, you must specify a fully-qualified G0000V00 name, you should include a DD statement for the GDG base name, to provide data set integrity on that base.

For information about generation numbers, see *DFSMS/MVS Using Data Sets*.

Relative Generation Numbers

When creating a generation data set, the relative generation number tells the system whether this is the first data set being added during the job, the second, the third, etc. When retrieving a generation data set, the relative generation number tells the system how many data sets have been added to the group since this data set was added.

The first time you use a relative generation number for a generation data group within a job, the system establishes the relationship between the relative generation number and the absolute generation number. The system maintains this relationship throughout the job.

For example, if you create a generation data set with a relative generation number of (+1), the system recognizes any subsequent reference to (+1) throughout the job as having the same absolute generation number.

Relative generation numbers are obtained from the catalog as it existed:

- **For JES2**, at the beginning of the first step that specifies the generation data set by relative generation number.

© Copyright IBM Corp. 1988, 1999

Note: In a shared DASD environment, if two or more jobs running on different systems simultaneously create new generations of the same data set, one of the jobs could fail with a JCL error.

- **For JES3**, when the job is set up, and again by the system at the beginning of the first step that specifies the generation data set by relative generation number. If the most recent data set is not the same at both times, the results are unpredictable.

Types of SMS-Managed Data Sets in a GDG

An SMS-managed generation data group (GDG) can consist of cataloged sequential and direct data sets residing on direct access volumes. Generation data sets in a GDG can have like or unlike data set attributes and data set organizations. If a GDG is created on an SMS-managed volume, any dependencies on a model data set label in order to allocate a new generation data set should be removed. A GDG can contain both SMS-managed and non-SMS-managed generation data sets.

Types of Non-SMS-Managed Data Sets in a GDG

A non-SMS-managed generation data group (GDG) can consist of cataloged sequential and direct data sets residing on tape volumes, direct access volumes, or both. Generation data sets in a GDG can have like or unlike DCB attributes and data set organizations.

Retrieval of GDG Data Sets

All of the generations of a generation data group can be retrieved together as a single data set. The retrieval order is last-in-first-out.

Building a GDG Base Entry

Before creating the first generation data set, build a generation data group base entry in a VSAM, OS CVOL, or integrated catalog facility catalog. This base entry must provide for as many generation data sets, up to 255, as you would like to have in the GDG. The system uses the base to keep track of the chronological order of the generation data sets.

Use the access method services DEFINE command to build generation data group bases in a VSAM or integrated catalog facility catalog. This command is described in *DFSMS/MVS Access Method Services for ICF*, or *DFSMS/MVS Access Method Services for VSAM*.

Defining Attributes for SMS-Managed Generation Data Sets

Data Class and Storage Class

Another requirement (in addition to a GDG base entry) for an SMS-managed GDG is a storage class for a new generation data set. The system uses the attributes defined in the data class and storage class when you create a new generation data set.

Note: Rather than using a data class to specify data set allocation attributes, you can specify the LIKE or the REFDD parameter.

You can let the installation-written automatic class selection (ACS) routines select a data class and storage class for a new generation data set, or you can specify the DATACLAS and STORCLAS parameters on the DD statement. Also, you can specify those DD parameters that override attributes in the data class and storage class (such as RECORG, LRECL, SPACE, and so on). See the DATACLAS and STORCLAS DD parameters in *OS/390 MVS JCL Reference*.

Creating an SMS-Managed Generation Data Set

When creating a new SMS-managed generation data set, always code the DSNAME and DISP parameters and optionally, code the DATACLAS and STORCLAS parameters.

DSNAME Parameter

In the DSNAME parameter, code the name of the GDG followed by a number, +1 to +255, in parentheses. If this is the first data set being added to a GDG in the job, code +1 in parentheses. Each time in the job you add a data set to the same GDG, increase the number by one.

When referring to this data set in a subsequent job step, code the relative generation number used to create it on the DSNAME parameter. You cannot refer to this data set in the step in which it was created. At the end of the job, the system updates the relative generation numbers of all generations in the group to reflect the additions.

Note: If the relative generation number makes the absolute generation number exceed G9999Vyy, wraparound occurs. In an integrated catalog facility catalog, if you create a new generation data set with a relative generation number, such as (+1), and an absolute generation number of G9999Vyy exists in the GDG base, the wraparound generates number G0001Vyy. In a VSAM catalog, a wraparound generates number G0000Vyy and unpredictable results can occur. (For information about absolute generation numbers and version numbers, in the form GxxxxVyy, see *DFSMS/MVS Using Data Sets*.)

DATACLAS and STORCLAS Parameters

If the ACS routines do not select the needed data class or storage class, code the DATACLAS or STORCLAS parameters (and any DD parameters needed to override attributes in the data class or storage class).

Disposition of SMS-Managed Generation Data Sets

New SMS-managed generation data sets are cataloged in a deferred roll-in status when created. This means that they are temporarily cataloged by their GxxxxVyy number but an entry is not made in the GDG base at this time. Then at step termination, the generations are processed depending on their normal termination disposition as described in the following paragraphs. (For information about absolute generation numbers and version numbers, in the form GxxxxVyy, see *DFSMS/MVS Using Data Sets*.)

DISP Parameter

Assign new generation data sets a status of NEW and a normal termination disposition of CATLG, KEEP, DELETE, or PASS.

DISP=(NEW,CATLG)

At step termination, the deferred generation data set is rolled into the GDG base. This means that the temporary catalog entry is removed and an entry is made in the GDG base.

DISP=(NEW,KEEP)

At step and job termination, the deferred generation data set remains in a deferred roll-in state. This means that the temporary catalog entry is not removed and an entry is not made in the GDG base.

DISP=(NEW,DELETE)

At step termination, the deferred generation data set is scratched and uncataloged.

DISP=(NEW,PASS)

At job termination, the deferred generation data set is scratched and uncataloged.

Note: If you create a new generation data set and a deferred generation data set exists with the same GxxxxVyy number, the number and its associated space are reused.

Defining Attributes for Non-SMS-Managed Generation Data Sets

Another requirement (in addition to a GDG base entry) for a GDG is a data set label. The system uses this label to refer to DCB attributes and the EXPDT value when you create a new generation data set.

DCB attributes can be supplied in one of the following ways:

1. Create a model data set label on the volume on which the index resides (the volume containing the GDG base)
2. Refer to a cataloged data set to use its attributes
3. Specify LIKE= or REFDD= to use attributes from the DD statement or specify DATACLAS to use attributes specified for the data class

Attributes can be supplied before you catalog a generation, when you catalog it, or at both times, as follows:

1. Create a model data set label on the volume on which your index resides. You can provide initial DCB attributes when you create your model; however, you need not provide any attributes at this time. Because only the attributes in the data set label are used, the model data set can be allocated with SPACE=(TRK,0) to conserve direct access space. (For an indexed sequential data set, a space request greater than 0 is required.) Initial or overriding attributes can be supplied when you create and catalog a generation.

   To create a model data set label, include the following DD statement in the job step that builds the index or in any other job step that precedes the step in which you create and catalog your generation.

```
//name  DD  DSNAME=datagrpname,DISP=(,KEEP),SPACE=(TRK,0),
//          UNIT=yyyy,VOLUME=SER=xxxxxx,
//          DCB=(applicable subparameters)
```

The DSNAME is the common name by which each generation is identified; therefore, the model data set label cannot be cataloged. The GDG base is an entity that resides in the catalog. xxxxxx is the serial number of the volume containing the catalog where the GDG base resides. The applicable DCB subparameters for a model data set label are DSORG, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP. If no DCB subparameters are wanted initially, you need not code the DCB parameter.

2. You do not need to create a model data set label if either of the following is true:

   a. You can refer to a cataloged data set with attributes identical to those you want or to an existing model data set label for which you can supply overriding attributes.

   b. The DCB attributes are supplied by the specified or selected data class.

   To refer to a cataloged data set for the use of its attributes, specify DCB=dsname on the DD statement that creates and catalogs your generation.

   To refer to an existing model, specify DCB=(modeldscbname,attributes) on the DD statement that creates and catalogs your generation. With SMS, specify LIKE=modeldsname or REFDD=*.ddname, *.stepname.ddname, or *.stepname.procstepname.ddname to refer to an earlier DD statement that identifies the model data set name. For more information, see “Modeling Data Set Attributes” on page D-4.

   To specify a data class, code DATACLAS=dataclass on the DD statement (although system ACS routines might override the value you code) or use the system default. For more information about data class, see “Specifying Constructs” on page D-2.

Creating a Non-SMS-Managed Generation Data Set

When creating a new non-SMS-managed generation data set, always code the DSNAME, DISP, and UNIT parameters and optionally, code the VOLUME, SPACE, LABEL, and DCB parameters.

DSNAME Parameter

In the DSNAME parameter, code the name of the GDG followed by a number, +1 to +255, in parentheses. If this is the first data set being added to a GDG in the job, code +1 in parentheses. Each time in the job you add a data set to the same GDG, increase the number by one.

When referring to this data set in a subsequent job step, code the relative generation number used to create it on the DSNAME parameter. You cannot refer to this data set in the step in which it was created. At the end of the job, the system updates the relative generation numbers of all generations in the group to reflect the additions.

Note: If the relative generation number makes the absolute generation number exceed G9999Vyy, wraparound occurs. In an integrated catalog facility catalog, if you create a new generation data set with a relative generation number, such as (+1), and an absolute generation number of G9999Vyy

exists in the GDG base, the wraparound generates number G0001Vyy. In a VSAM catalog, a wraparound generates number G0000Vyy and unpredictable results can occur.

(For information about absolute generation numbers and version numbers, in the form GxxxxVyy, see *DFSMS/MVS Using Data Sets*.)

DISP Parameter

Assign new generation data sets a status of new and a disposition of catalog: DISP=(NEW,CATLG).

UNIT Parameter

The UNIT parameter is required for a new generation data set unless VOLUME=REF=reference is coded. In the UNIT parameter, identify the type of device wanted.

VOLUME Parameter

Assign a volume in the VOLUME parameter, or omit the VOLUME parameter and let the system assign the volume. The VOLUME parameter can request a private volume, PRIVATE, and more than one volume in the volume count.

SPACE Parameter

Code the SPACE parameter when the generation data set is to reside on a direct access volume.

LABEL Parameter

You can specify label type; password protection, PASSWORD; and a retention period, EXPDT or RETPD, in the LABEL parameter. If the data set is to reside on a tape volume and is not the first data set on the volume, specify a data set sequence number.

DCB Parameter

If you use a model data set label from the same GDG and if the label contains all the attributes for this generation data set, omit the DCB parameter. If all the attributes are not contained in the label or if you want to override certain attributes, code DCB=(list of attributes).

If you use a model data set label from a different GDG and if the label contains all the attributes for this generation data set, code DCB=dsname. If some attributes are missing from the label or if you want to override some attributes, code DCB=(dsname,list of attributes).

If a model data set label does not exist, you must use the label for a cataloged data set. Code DCB=dsname. If some attributes are missing from the label, or if you want to override some attributes, code DCB=(dsname,list of attributes).

Retrieving a Generation Data Set

To retrieve an SMS-managed generation data set, always code the DSNAME and DISP parameters.

To retrieve a non-SMS-managed generation data set, always code the DSNAME and DISP parameters. Optional parameters are the UNIT, VOLUME, LABEL, and DCB.

DSNAME Parameter

For both SMS-managed and non-SMS-managed data sets, use the DSNAME parameter to retrieve a single generation data set or all of the generation data sets in the GDG.

Retrieving a Single Generation Data Set

To retrieve a single generation data set, code in the DSNAME parameter the name of the generation data group followed by a relative generation number in parentheses. The number indicates which generation data set is to be retrieved. To retrieve the most recent data set, code a zero.

To retrieve data sets created before the most recent data set, code a minus value, -1 to -255. The value of nnn indicates the relation of the desired data set to the most current data set: (-1) refers to the data set created immediately before the most recent data set; (-2) refers to the data set created before the data set identified by (-1).

For example:

PAYROLL	Name of the GDG
DSNAME=PAYROLL(0)	This week's generation data set
DSNAME=PAYROLL(-1)	Last week's generation data set
DSNAME=PAYROLL(-2)	Generation data set of two weeks ago

Relative generation numbers are maintained by the system only when generation data sets are specified using relative generation numbers.

Note: Refer to generation data sets in a deferred roll-in state by their relative number, such as (+1), within the job that creates it. Refer to generation data sets in a deferred roll-in state by their absolute generation number (GxxxxVyy) in subsequent jobs. For more information on how to refer to GDG data sets in a deferred roll-in state, see *DFSMS/MVS Using Data Sets*.

Note: When retrieving a generation data set within a started task, and the generation data set is cataloged in a private catalog or control volume (CVOL), coding a relative generation number produces unpredictable results.

Retrieving All Generation Data Sets

To retrieve all generations of a GDG as a single data set, specify the GDG name without a generation number in the DSNAME parameter; this is called a GDG ALL request. For example:

DSNAME=PAYROLL For all generations

To use a GDG ALL request, the DCB attributes and data set organization of all generations must be identical.

The system treats a GDG ALL request as a concatenation of all existing data sets in the GDG, starting with the most recent data set and ending with the oldest, which can affect the meaning of system messages in the job output listing.

For example, assume that data set GDGDS has two generations and that data sets A and B are not generation data sets. To concatenate A, all generations of GDGDS, and B, you code the following JCL:

```
//DD1      DD  DSN=A,DISP=SHR
//         DD  DSN=GDGDS,DISP=SHR
//         DD  DSN=B,DISP=SHR
```

Because of the GDG ALL request, the system treats DD1 as if you had coded the following statements, and assigns the following relative position numbers:

```
//DD1      DD  DSN=A,DISP=SHR             +000
//         DD  DSN=GDGDS(0),DISP=SHR      +001
//         DD  DSN=GDGDS(-1),DISP=SHR     +002
//         DD  DSN=B,DISP=SHR             +003
```

Any error message uses the relative position based on each generation included, not the position you explicitly specified. For example, an error message that includes a relative position of +002 refers to GDGDS(-1), not data set B.

All older generations have unit affinity to the newest data set.

For a GDG on tape, when you use a GDG ALL request and specify parallel mounting in the UNIT parameter, the system mounts all volumes of only the **first** generation.

For a GDG on direct access, when you use a GDG ALL request and specify parallel mounting in the UNIT parameter, the system mounts all volumes of **all** generations.

DISP Parameter

For both SMS-managed and non-SMS-managed data sets, always code the DISP parameter. The first subparameter of the DISP parameter must be OLD, SHR, or MOD. If you code MOD for a generation data set and the specified relative generation does not exist in the catalog, the system changes the status to NEW.

A normal termination disposition is optional when retrieving a generation data set but is required in a GDG ALL request. Do not code PASS in a GDG ALL request.

UNIT Parameter

For non-SMS-managed data sets, code the unit-count subparameter in the UNIT parameter when you want more than one device assigned to the data set. Or, if the data set resides on more than one volume and you want as many devices as there are volumes, code P in the UNIT parameter.

VOLUME Parameter

For non-SMS-managed data sets, use the VOLUME parameter to request a private volume, PRIVATE, and to indicate that more volumes might be required, volume count. For an old generation data set, do not specify either a volume serial number or a volume reference to another data set or to an earlier DD statement.

LABEL Parameter

For non-SMS-managed data sets, code the LABEL parameter when the data set is on tape and has other than standard labels. If the data set is not the first data set on the volume, specify the data set sequence number. If the data set sequence number is coded for a GDG ALL request, it is ignored; the data set sequence number is obtained from the catalog.

DCB Parameter

For non-SMS-managed data sets, code DCB=(list of attributes) when the data set has other than standard labels and DCB information is required to complete the data control block. Do not code DCB=dsname.

Deleting and Uncataloging Generation Data Sets

Note that uncataloging is not supported for SMS-managed data sets.

In a multiple-step job, catalog or uncatalog generation data sets using the DD DISP parameter. Do not use the IEHPROGM utility program or a user program. Because system routines access the catalog during job execution, they are unaware of the functions performed by IEHPROGM or a user program; you might get unpredictable results.

If a DD statement in a multiple-step job tries to delete or uncatalog any generation data set **except** the oldest in a GDG, catalog management can lose orientation within the data group. This could cause the deletion, uncataloging, or retrieval of the wrong data set when you later refer to a specific generation. Therefore, if you delete a generation data set in a multiple-step job, do not refer to any older generations in later job steps.

When you delete a generation data group in a multiple-step job, remember that the first time you use a relative generation number for a generation data group within a job, the system establishes the relationship between the relative generation number and the absolute generation number. The system maintains this relationship throughout the job.

The following examples illustrate how the system maintains this relationship when deleting a generation data group:

Assume the following generation data sets already exist with absolute generation numbers: G0006V00, G0007V00, and G0008V00.

Issue the following JCL:

```
//STEP1  EXEC
//DD1    DD    DISP=OLD,DSN=A.B.C(-1)
//STEP2  EXEC
//DD2    DD    DISP=(OLD,DELETE),DSN=A.B.C
//STEP3  EXEC
//DD3    DD    DISP=(NEW,CATLG),DSN=A.B.C(+1)
```

In the above example, the absolute generation number referenced by relative generation number in STEP1 (DD1) is G0007V00. The system establishes the relative/absolute relationship that it will maintain throughout the job. In STEP2, all generation data sets are to be deleted, which occurs at STEP2 termination. In STEP3, the system assigns the absolute generation number G0009V00 to the new generation data set created (DD3).

In the following example, the JCL is set up to delete all generation data sets at the beginning of the job.

```
//STEP1  EXEC
//DD1    DD    DISP=(OLD,DELETE),DSN=A.B.C
//STEP2  EXEC
//DD2    DD    DISP=(NEW,CATLG),DSN=A.B.C(+1)
//STEP3  EXEC
//DD3    DD    DISP=(NEW,CATLG),DSN=A.B.C(+2)
```

In this second example, the system establishes the relative/absolute relationship in STEP2, the first time that a relative generation number is used in the job. The system then assigns absolute generation number G0001V00 to the data set referenced in DD2 and absolute generation number G0002V00 to the data set referenced in DD3.

Submitting a Job for Restart

Certain rules apply when you refer to generation data sets in a job submitted for restart using the RESTART parameter on the JOB statement.

For Step Restart

To refer to generation data sets that were created and cataloged in steps before the restart step, use their present relative generation numbers. For example, if the last generation data set created and cataloged was assigned a generation number of +2, it would be referred to as 0 in the restart step and in steps following the restart step. In this case, the generation data set assigned number of +1 when created would be referred to as -1.

For Checkpoint Restart

If generation data sets created in the restart step were kept instead of cataloged, that is, DISP=(NEW,CATLG,KEEP), you can, during checkpoint restart, refer to these data sets and generation data sets created and cataloged in steps before the restart step by the same relative generation numbers that were used to create them.

For Deferred Checkpoint Restart

The system does not use the catalog to obtain the volume serial numbers for a GDG. Therefore, if you changed the volume serial numbers in the catalog between the original submission of the job and the restart, you **must** code volume serial numbers.

Example 1

For SMS-managed data sets:

```
//STEPA  EXEC PGM=PROCESS
//DD1    DD   DSNAME=A.B.C(+1),DISP=(NEW,CATLG)
//DD2    DD   DSNAME=A.B.C(+2),DISP=(NEW,CATLG)
//DD3    DD   DSNAME=A.B.C(+3),DISP=(NEW,CATLG)
```

This step shows the DD statements used to add three SMS-managed data sets to a GDG.

The installation-written automatic class selection (ACS) routines are used to select a data class and storage class for the data sets.

Example 2

For SMS-managed data sets:

```
//JWC    JOB  ,'J. GRIFFIN-KEENE'
//STEP1  EXEC PGM=REPORT9
//DDA    DD   DSNAME=A.B.C(-2),DISP=OLD
//DDB    DD   DSNAME=A.B.C(-1),DISP=OLD
//DDC    DD   DSNAME=A.B.C(0),DISP=OLD
```

This job shows the DD statements needed to retrieve the SMS-managed generation data sets created in the first example, when the GDG contains no other generation data sets.

Example 3

For non-SMS-managed data sets:

```
//STEPA  EXEC PGM=PROCESS
//DD1    DD   DSNAME=A.B.C(+1),DISP=(NEW,CATLG),UNIT=3400-6,
//            VOL=SER=13846,LABEL=(,SUL)
//DD2    DD   DSNAME=A.B.C(+2),DISP=(NEW,CATLG),UNIT=3330,
//            VOL=SER=10311,SPACE=(480,(150,20))
//DD3    DD   DSNAME=A.B.C(+3),DISP=(NEW,CATLG),UNIT=3350,
//            VOL=SER=28929,SPACE=(480,(150,20)),
//            DCB=(LRECL=120,BLKSIZE=480)
```

This step shows the DD statements used to add three non-SMS-managed data sets to a GDG.

DD1 and DD2 do not include the DCB parameter because a model data set label exists on the same volume as the GDG index and has the same name as the GDG: A.B.C. Because the DCB parameter is coded on the third DD statement, the attributes LRECL and BLKSIZE override the attributes included in the model data set label.

Example 4

For non-SMS-managed data sets:

```
//JWC    JOB  ,'J. GRIFFIN-KEENE'
//STEP1  EXEC PGM=REPORT9
//DDA    DD   DSNAME=A.B.C(-2),DISP=OLD,LABEL=(,SUL)
//DDB    DD   DSNAME=A.B.C(-1),DISP=OLD
//DDC    DD   DSNAME=A.B.C(0),DISP=OLD
```

This job shows the DD statements needed to retrieve the non-SMS-managed generation data sets created in the third example, when the GDG contains no other generation data sets.

Example 5

For SMS-managed data sets:

```
//J1   JOB   ACCT34,'DEPT.17'
//S11  EXEC  PGM=P1
//A    DD    DSNAME=GDGDS(+1),DISP=(NEW,CATLG),STORCLAS=...
//S12  EXEC  PGM=P2
//B    DD    DSNAME=GDGDS(+2),DISP=(NEW,CATLG),STORCLAS=...
//S13  EXEC  PGM=P3
//C    DD    DSNAME=GDGDS(+1),DISP=OLD
       .
       .
//J2   JOB   ACCT34,'DEPT.17'
//S21  EXEC  PGM=P4
//D    DD    DSNAME=GDGDS,DISP=OLD
//S22  EXEC  PGM=P5
//E    DD    DSNAME=GDGDS(0),DISP=OLD
//S23  EXEC  PGM=P6
//F    DD    DSNAME=GDGDS(+1),DISP=(NEW,CATLG),STORCLAS=...
//S24  EXEC  PGM=P7
//G    DD    DSNAME=GDGDS(+2),DISP=(NEW,CATLG),STORCLAS=...
//S25  EXEC  PGM=P8
//H    DD    DSNAME=GDGDS(+1),DISP=OLD
//S26  EXEC  PGM=P9
//J    DD    DSNAME=GDGDS(+2),DISP=OLD
//S27  EXEC  PGM=P10
//K    DD    DSNAME=GDGDS(0),DISP=OLD
//S28  EXEC  PGM=P11
//L    DD    DSNAME=GDGDS(-1),DISP=OLD
//S29  EXEC  PGM=P12
//M    DD    DSNAME=GDGDS,DISP=OLD
       .
       .
```

These two jobs show the creation and retrieval of generation data sets.

DD statement A - create 1st generation (cataloged at allocation, rolled in at end of step).
DD statement B - create 2nd generation (cataloged at allocation, rolled in at end of step).
DD statement C - reference 1st generation.
At the end of job J1, generation 1 and 2 have been cataloged.

DD statement D - reference all generations (1st and 2nd).
DD statement E - reference 2nd generation.
DD statement F - create 3rd generation (cataloged at allocation, rolled in at end of step).
DD statement G - create 4th generation (cataloged at allocation, rolled in at end of step).
DD statement H - reference 3rd generation.
DD statement J - reference 4th generation.
DD statement K - reference 2nd generation.

DD statement L - reference 1st generation.
DD statement M - reference all generations (1st through 4th).
At the end of job J2, generation 3 and 4 have been cataloged.

Example 6

For non-SMS-managed data sets:

```
//J1   JOB   ACCT34,'DEPT.17'
//S11  EXEC  PGM=P1
//A    DD    DSNAME=GDGDS(+1),DISP=(,CATLG),UNIT=...
//S12  EXEC  PGM=P2
//B    DD    DSNAME=GDGDS(+2),DISP=(,CATLG),UNIT=...
//S13  EXEC  PGM=P3
//C    DD    DSNAME=GDGDS(+1),DISP=OLD
       .
       .
//J2   JOB   ACCT34,'DEPT.17'
//S21  EXEC  PGM=P4
//D    DD    DSNAME=GDGDS,DISP=OLD
//S22  EXEC  PGM=P5
//E    DD    DSNAME=GDGDS(0),DISP=OLD
//S23  EXEC  PGM=P6
//F    DD    DSNAME=GDGDS(+1),DISP=(,CATLG),UNIT=...
//S24  EXEC  PGM=P7
//G    DD    DSNAME=GDGDS(+2),DISP=(,CATLG),UNIT=...
//S25  EXEC  PGM=P8
//H    DD    DSNAME=GDGDS(+1),DISP=OLD
//S26  EXEC  PGM=P9
//J    DD    DSNAME=GDGDS(+2),DISP=OLD
//S27  EXEC  PGM=P10
//K    DD    DSNAME=GDGDS(0),DISP=OLD
//S28  EXEC  PGM=P11
//L    DD    DSNAME=GDGDS(-1),DISP=OLD
//S29  EXEC  PGM=P12
//M    DD    DSNAME=GDGDS,DISP=OLD
       .
       .
```

These two jobs show the creation and retrieval of generation data sets.

DD statement A - create 1st generation (and catalog at end of step).
DD statement B - create 2nd generation (and catalog at end of step).
DD statement C - reference 1st generation.
At the end of job J1, generation 1 and 2 have been cataloged.

DD statement D - reference all generations (1st and 2nd).
DD statement E - reference 2nd generation.
DD statement F - create 3rd generation (and catalog at end of step).
DD statement G - create 4th generation (and catalog at end of step).
DD statement H - reference 3rd generation.
DD statement J - reference 4th generation.
DD statement K - reference 2nd generation.
DD statement L - reference 1st generation.
DD statement M - reference all generations (1st through 4th).
At the end of job J2, generation 3 and 4 have been cataloged.

Appendix C. VSAM Data Sets

Virtual storage access method (VSAM) can be used for data sets on direct access storage. Because VSAM is different from the other access methods, certain DD parameters and subparameters are different for VSAM data sets.

This appendix has two main topics:

- **VSAM Data Sets - With SMS**. Use this topic if SMS is installed and active, see topic C-1.
- **VSAM Data Sets - Without SMS**. Use this topic if SMS is not installed or is not active, see topic C-4.

VSAM Data Sets - With SMS

Creating a VSAM Data Set - With SMS

To create a permanent VSAM data set, either (1) use access method services commands or (2) use the DD statement with the RECORG parameter or with a data class that contains RECORG. You can also create temporary VSAM data sets.

See Appendix D, Data Sets with SMS for information about SMS.

To create a VSAM data set, code a DD statement in the form:

```
//ddname DD  DSNAME=dsname,RECORG=record-organization,
//         DISP=(NEW,...)...
```

If a data class contains RECORG, code the DD statement as:

```
//ddname DD  DSNAME=dsname,DATACLAS=data-class-name,
//         DISP=(NEW,...)...
```

The system catalogs a permanent VSAM data set when the data set is allocated.

Retrieving an Existing VSAM Data Set - With SMS

To retrieve an existing VSAM data set, code a DD statement in the form:

```
//ddname DD  DSNAME=dsname,DISP=OLD
//ddname DD  DSNAME=dsname,DISP=SHR
```

You can pass VSAM data sets within a job. (Note that the system replaces PASS with KEEP for old permanent VSAM data sets. When you refer to the data set later in the job, the system obtains data set information from the catalog.)

Migration Consideration for SMS

If you have existing JCL that allocates a VSAM data set with DISP=(OLD,DELETE), note that without SMS, the system ignores DELETE and keeps the data set. However, with SMS, DELETE is valid and the system deletes the data set.

© Copyright IBM Corp. 1988, 1999

DD Statement Parameters - With SMS

The DD statement parameters that you should use for VSAM data sets are shown in Figure C-1. You can also use other DD parameters, if needed, to override attributes in the data class (such as KEYLEN and KEYOFF). Parameters that should not be used or should be used only with caution are explained in Figure C-2 on page C-3.

Figure C-1 (Page 1 of 2). With SMS, DD Parameters to Use when Processing VSAM Data Sets

Parameter	Subparameter	Comment
AMP		This parameter has subparameters for: 1. Overriding operands specified with the ACB, EXLST, or the GENCB macro instructions 2. Supplying operands missing from the ACB or GENCB macro instruction 3. Indicating checkpoint/restart options 4. Indicating options when using ISAM macro instructions to process a key-sequenced data set 5. Indicating that the data set is a VSAM data set when the DD statement specifies unit and volume information or DUMMY 6. Indicating that VSAM is to supply storage dumps of the ACBs that identify the DD statement
DATACLAS	*data-class-name*	No special considerations for VSAM data sets, except that the record organization (RECORG) should specify a VSAM data set.
DDNAME	*ddname*	No special considerations for VSAM data sets.
DISP	all subparameters	All DISP subparameters can be used for VSAM data sets except UNCATLG, which is ignored (KEEP is implied if UNCATLG is coded).
DSNAME	*dsname*	VSAM uses the dsname. An area-name (area-name), generation number (generation), or member name (member) is ignored if coded with dsname.
	All temporary *dsnames*	You can code a temporary dsname for a VSAM data set.
	All backward DD references of the form *.ddname	You can code backward references to VSAM data sets on the REFDD parameter.
DUMMY		No special considerations for VSAM, except that an attempt to read results in an end-of-data condition, and an attempt to write results in a return code that indicates the write was successful. If specified, AMP=AMORG must also be specified.
DYNAM		No special considerations for VSAM data sets.
FREE		No special considerations for VSAM data sets.
MGMTCLAS	*mgmt-class-name*	No special considerations for VSAM data sets.
RECORG	**KS**	Specifies a key-sequenced data set.
	ES	Specifies an entry-sequenced data set.
	RR	Specifies a relative record data set.
	LS	Specifies a linear space data set.
		RECORG overrides the record organization in the data class.
SECMODEL		No special considerations for VSAM data sets.
SPACE		No special considerations for VSAM data sets.
STORCLAS	*storage-class-name*	No special considerations for VSAM data sets. If a storage class is assigned, the VSAM data set is managed by SMS.

Figure C-1 (Page 2 of 2). With SMS, DD Parameters to Use when Processing VSAM Data Sets

Parameter	Subparameter	Comment
UNIT	*device number*	Must be the device number of a valid device for VSAM (2305, 3330, 3340, 3344, 3350, 3375, 3380, 3390, or 9345). If not, OPEN will fail.
	type	Must be a type supported by VSAM (2305, 3330, 3340, 3350, 3375, 3380, 3390, or 9345). If not, OPEN will fail.
	group	Must be a group supported by VSAM. If not, OPEN will fail.
	p	System must have enough units to mount all of the volumes specified. If sufficient units are available, UNIT=P can improve performance by avoiding the mounting and demounting of volumes. For multivolume data sets defined in integrated catalog facility catalogs, UNIT=(,P) must be specified because all primary volumes must be mounted in parallel.
	unit count	If the number of devices requested is greater than the number of volumes on which the data set resides, the extra devices are allocated anyway. If a key-sequenced data set and its index reside on unlike devices, the extra devices are allocated evenly between the unlike device types. If the number of devices requested is less than the number of volumes on which the data set resides but greater than the minimum number required to gain access to the data set, the devices over the minimum are allocated evenly between unlike device types. If devices beyond the count specified are in use by another task but can be shared and have mounted on them volumes containing parts of the data set to be processed, they will also be allocated to this data set.
	DEFER	No special considerations for VSAM.
Note: MVS/ESA SP 5.2 does not support multiple exposure devices, such as the 2305, 3350P, and 3351P.		
VOLUME	**PRIVATE**	No special considerations for VSAM.
	SER	The volume serial number(s) used in the access method services DEFINE command for the data set must match the volume serial numbers in the VOLUME=SER specification when the data set is defined. After a VSAM data set is defined, the volume serial number(s) need not be specified on a DD statement to retrieve the data set. If, however, VOLUME=SER and UNIT=type are specified, only those volumes specifically named are initially mounted. Other volumes may be mounted when needed, if at least one of the units allocated to the data set cannot be shared or the unit count is equal to the total number of volumes allocated to the data set. A unit cannot be shared when the unit count is less than the number of volume serial numbers specified or when DEFER is specified. If VOLUME=SER is specified and the data set is cataloged in a user catalog, the user catalog should be defined as a JOBCAT or a STEPCAT for the current step.

Figure C-2 (Page 1 of 2). With SMS, DD Parameters to Avoid when Processing VSAM Data Sets

Parameter	Subparameter	Comment
BURST		Not applicable.
CHARS		Not applicable.
CHKPT		VSAM ignores CHKPT.
COPIES		Not applicable.
DATA		Not applicable.
DCB	All	Not applicable.
DEST		Specify DEST only with the SYSOUT parameter.
DLM		Not applicable.
FCB		Not applicable.
FLASH		Not applicable.

Figure C-2 (Page 2 of 2). With SMS, DD Parameters to Avoid when Processing VSAM Data Sets

Parameter	Subparameter	Comment
LABEL	**BLP, NL, NSL**	Not applicable
	IN	Not applicable
	OUT	Not applicable
	NOPWREAD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	PASSWORD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	SL, SUL	Although these parameters apply to direct-access storage devices, SL is always used for VSAM, whether you specify SL, SUL, or neither.
MODIFY		Not applicable.
SYSOUT		If SYSOUT is coded with a mutually exclusive parameter (for example, DISP), the job step is terminated with an error message.
UCS	All	Not applicable.
UNIT	**AFF**	Use this subparameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of UNIT=AFF are unpredictable.
VOLUME	**REF**	Use this subparameter carefully. If the referenced volumes are not a subset of those contained in the catalog record for the data set, the results are unpredictable.
	vol-seq-number	Results are unpredictable.
	volume-count	Not applicable because this subparameter gives the number of nonspecific volumes. All VSAM volumes must be specifically defined.
*	Not applicable.	

VSAM Data Sets - Without SMS

Creating a VSAM Data Set - Without SMS

Use access method services commands to create a VSAM data set. You cannot use a DD statement.

Retrieving an Existing VSAM Data Set - Without SMS

To request a cataloged VSAM data set, code a DD statement in the form:

```
//ddname DD  DSNAME=dsname,DISP=OLD
//ddname DD  DSNAME=dsname,DISP=SHR
```

Note: VSAM data sets cannot be passed within a job.

DD Statement Parameters - Without SMS

DD statement parameters that can be used without modification are explained in Figure C-3. Parameters that should not be used or should be used only with caution are explained in Figure C-4 on page C-6.

VSAM has one DD statement parameter of its own, AMP. The AMP parameter takes effect when the data set defined by the DD statement is opened.

Figure C-3 (Page 1 of 2). Without SMS, DD Parameters to Use when Processing VSAM Data Sets

Parameter	Subparameter	Comment
AMP		This parameter has subparameters for: 1. Overriding operands specified with the ACB, EXLST, or the GENCB macro instructions 2. Supplying operands missing from the ACB or GENCB macro instruction 3. Indicating checkpoint/restart options 4. Indicating options when using ISAM macro instructions to process a key-sequenced data set 5. Indicating that the data set is a VSAM data set when the DD statement specifies unit and volume information or DUMMY 6. Indicating that VSAM is to supply storage dumps of the ACBs that identify the DD statement
DDNAME	*ddname*	No special considerations for VSAM.
DISP	**SHR**	Indicates that you are willing to share the data set with other jobs. This subparameter alone, however, does not guarantee that sharing will take place. See *DFSMS/MVS Using Data Sets* for a full description of data-set sharing.
	OLD	No special considerations for VSAM.
DSNAME	*dsname*	Names the VSAM cluster to which the data set belongs.
DUMMY		No special considerations for VSAM, except that an attempt to read results in an end-of-data condition, and an attempt to write results in a return code that indicates the write was successful. If specified, AMP=AMORG must also be specified.
DYNAM		No special considerations for VSAM.
FREE		No special considerations for VSAM.
PROTECT		No special considerations for VSAM.
UNIT	*device number*	Must be the device number of a valid device for VSAM (2305, 3330, 3340, 3344, 3350, 3375, 3380, 3390, or 9345). If not, OPEN will fail
	type	Must be a type supported by VSAM (2305, 3330, 3340, 3350, 3375, 3380, 3390, or 9345). If not, OPEN will fail.
	group	Must be a group supported by VSAM. If not, OPEN will fail.
	p	System must have enough units to mount all of the volumes specified. If sufficient units are available, UNIT=P can improve performance by avoiding the mounting and demounting of volumes. For multivolume data sets defined in integrated catalog facility catalogs, UNIT=(,P) must be specified because all primary volumes must be mounted in parallel.
	unit count	If the number of devices requested is greater than the number of volumes on which the data set resides, the extra devices are allocated anyway. If a key-sequenced data set and its index reside on unlike devices, the extra devices are allocated evenly between the unlike device types. If the number of devices requested is less than the number of volumes on which the data set resides but greater than the minimum number required to gain access to the data set, the devices over the minimum are allocated evenly between unlike device types. If devices beyond the count specified are in use by another task but can be shared and have mounted on them volumes containing parts of the data set to be processed, they will also be allocated to this data set.
	DEFER	No special considerations for VSAM.

Note: MVS/ESA SP 5.2 does not support multiple exposure devices, such as the 2305, 3350P, and 3351P.

Figure C-3 (Page 2 of 2). Without SMS, DD Parameters to Use when Processing VSAM Data Sets

Parameter	Subparameter	Comment
VOLUME	**PRIVATE**	No special considerations for VSAM.
	SER	The volume serial number(s) used in the access method services DEFINE command for the data set must match the volume serial numbers in the VOLUME=SER specification when the data set is defined. After a VSAM data set is defined, the volume serial number(s) need not be specified on a DD statement to retrieve the data set. If, however, VOLUME=SER and UNIT=type are specified, only those volumes specifically named are initially mounted. Other volumes may be mounted when needed, if at least one of the units allocated to the data set cannot be shared or the unit count is equal to the total number of volumes allocated to the data set. A unit cannot be shared when the unit count is less than the number of volume serial numbers specified or when DEFER is specified. If VOLUME=SER is specified and the data set is cataloged in a user catalog, the user catalog should be defined as a JOBCAT or a STEPCAT for the current step.

Figure C-4 (Page 1 of 2). Without SMS, DD Parameters to Avoid when Processing VSAM Data Sets

Parameter	Subparameter	Comment
BURST		Not applicable.
CHARS		Not applicable.
CHKPT		VSAM ignores CHKPT.
COPIES		Not applicable.
DATA		Not applicable.
DCB	All	Not applicable.
DEST		Specify DEST only with the SYSOUT parameter.
DISP	**CATLG**	VSAM data sets are cataloged and uncataloged as a result of an access method services command; if CATLG is coded, a message is issued, but the data set is not cataloged.
	DELETE	VSAM data sets are deleted as a result of an access method services command; if DELETE is coded, a message is issued, but the data set is not deleted.
	MOD	For VSAM data sets, MOD is treated as if OLD were specified, except for processing with an ISAM program, in which case MOD indicates resume load.
	KEEP	Because KEEP is implied for VSAM data sets, it need not be coded.
	NEW	VSAM data spaces are initially allocated as a result of the access method services DEFINE command. If NEW is specified, the system allocates space, but it is never used by VSAM. Moreover, an access method services request for space may fail if the DISP=NEW acquisition of space causes too little space to remain available.
	UNCATLG	VSAM data sets are cataloged and uncataloged as a result of access method services commands; if UNCATLG is coded, a message is issued, but the data set is not uncataloged.
	PASS	Not applicable. However, because there is no error checking, coding PASS for a key-sequenced data set whose index resides on a like device does not result in an error. If a VSAM data set and its index reside on unlike devices, the results are unpredictable. In either case, the data set is not passed.
DLM		Not applicable.

Figure C-4 (Page 2 of 2). Without SMS, DD Parameters to Avoid when Processing VSAM Data Sets

Parameter	Subparameter	Comment
DSNAME	*dsname(area-name)* *dsname(generation)* *dsname(member)*	VSAM uses the dsname. An area-name, generation number, or member name is ignored, if coded with the dsname.
	All temporary *dsnames*	Do not code a temporary dsname for a VSAM data set.
	All backward DD references of the form **ddname*	Do not code backward references to VSAM data sets. If the object referred to is a cluster and the data set and index reside on unlike devices, the results of a backward DD reference are unpredictable.
FCB		Not applicable.
FLASH		Not applicable.
LABEL	**BLP, NL, NSL**	Not applicable.
	IN	Not applicable.
	OUT	Not applicable.
	NOPWREAD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	PASSWORD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	SL,SUL	Although these parameters apply to direct-access storage devices, SL is always used for VSAM, whether you specify SL, SUL, or neither.
MODIFY		Not applicable.
SPACE		VSAM data spaces are initially allocated as a result of the access method services DEFINE command. If SPACE is specified, the system allocates space, but it is never used by VSAM. Moreover, an access method services request for space may fail as a result of the SPACE acquisition of space.
SYSOUT		If SYSOUT is coded with a mutually exclusive parameter (for example, DISP), the job step is terminated with an error message.
UCS	All	Not applicable.
UNIT	**AFF**	Use this subparameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of UNIT=AFF are unpredictable.
VOLUME	**REF**	Use this subparameter carefully. If the referenced volumes are not a subset of those contained in the catalog record for the data set, the results are unpredictable.
	vol-seq-number	Results are unpredictable.
	volume-count	Not applicable because this subparameter gives the number of nonspecific volumes. All VSAM volumes must be specifically defined.
*	Not applicable.	

Appendix D. Data Sets with SMS

This appendix briefly summarizes information about defining data sets with SMS from this book and *OS/390 MVS JCL Reference*.

SMS, when installed and active, manages data sets and allows you to more easily define new data sets via DD statements. The storage administrator at your installation determines the data sets that are to be managed by SMS.

SMS can manage the following types of data sets:

- Physical sequential data sets
- Partitioned data sets (PDSs and PDSEs)
- VSAM data sets
- GDG data sets
- Temporary data sets
- VIO data sets

SMS **does not** manage the following types of data sets:

- Tape data sets
- ISAM data sets
- Sysout data sets (SYSOUT parameter)
- Subsystem data sets (SUBSYS parameter)
- TSO/E data sets coming from or going to a terminal
- TCAM message data sets
- Data sets on mass storage (MSS) volumes
- In-stream data sets

If the data set is to be managed through SMS, you cannot enclose the data set name in apostrophes on the DSNAME parameter on the DD statement. However, the following exception applies: You can enclose the data set name on the DSNAME parameter in apostrophes if the data set is to be assigned to, or already resides on, an SMS-managed mountable tape volume. This exception applies only if DFSMS/MVS 1.1 or later is installed.

Note: In this book, “with SMS” indicates information that applies when SMS is installed and active; “without SMS” indicates SMS is not installed or is not active.

SMS Constructs

With SMS, a new data set can have one or more of the following three constructs:

- Data class - contains the data set attributes related to the allocation of the data set.
- Management class - contains the data set attributes related to the migration and backup of the data set. A management class can only be assigned to a data set that also has a storage class assigned.
- Storage class - contains the data set attributes related to the storage occupied by the data set.

  A data set that has a storage class assigned is defined as an “SMS-managed data set”.

© Copyright IBM Corp. 1988, 1999

The storage administrator at your installation writes the automatic class selection (ACS) routines that SMS uses to assign the constructs to a new data set.

For example, with SMS

You can code the DDNAME, DSNAME, and DISP parameters to define a new data set:

```
//SMSDS0  DD  DSNAME=MYDS0.PGM,DISP=(NEW,KEEP)
```

and retrieve the data set with:

```
//SMSDSR  DD  DSNAME=MYDS0.PGM,DISP=MOD
```

In the example, installation-written ACS routines (possibly based on the data set name and information from your JOB, EXEC, and DD statements) can select a data class, management class, and storage class appropriate for the data set. You code only the ddname, dsname, and disposition of the data set. The constructs selected by the ACS routines contain all the other attributes needed to manage the data set.

Without SMS

You would have needed to code the data set attributes on the DCB, SPACE, UNIT, and VOLUME parameters; for example:

```
//SMSDS0  DD  DSNAME=MYDS0.PGM,VOLUME=SER=SYS084,
//          UNIT=SYSDA,SPACE=(TRK,(10,5)),DISP=(NEW,CATLG),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
```

Existing JCL

Generally, your existing JCL will continue to execute correctly. SMS allows the installation to benefit from the data class, management class, and storage class constructs without changing existing JCL. The installation-written ACS routines can be designed to filter existing parameters on the DD statement and select appropriate constructs for the data set.

Default Unit

Also with SMS, for a non-SMS-managed data set, if your storage administrator has set a system default unit under SMS, you do not need to specify UNIT. Check with your storage administrator.

Specifying Constructs

In many cases, the constructs selected by the installation-written ACS routines are sufficient for your data sets.

However, when defining a new data set, you can select a data class, management class, or storage class by coding one or more of the following DD parameters:

- DATACLAS - specifies the data class
- MGMTCLAS - specifies the management class
- STORCLAS - specifies the storage class

The storage administrator has defined the names of the classes you can specify. You can view the names and attributes defined in each of the named classes by using ISMF. See *DFSMS/MVS Using ISMF* for information on how to use ISMF.

For example, you can select the data class named DCLAS01 for a new data set with:

```
//SMSDS1  DD  DSNAME=MYDS1.PGM,DATACLAS=DCLAS01,DISP=(NEW,KEEP)
```

In the example, SMS uses the attributes in the data class named DCLAS01 to manage the data set. The installation-written ACS routines can select the management class and storage class.

Note that an ACS routine can override the data class, management class, or storage class that you specify.

Overriding Attributes Defined in the Data Class

For a new data set, you can, if needed, override the data class attributes defined in the data class for the data set by coding one or more of the following DD parameters:

RECORG (record organization) or RECFM (record format)
LRECL (record length)
KEYLEN (key length)
KEYOFF (key offset)
DSNTYPE (type, PDS or PDSE)
AVGREC (record request and space quantity)
SPACE (average record length, primary, secondary, and directory quantity)
RETPD (retention period) or EXPDT (expiration date)
VOLUME (volume-count)

For example:

```
//SMSDS2  DD  DSNAME=MYDS2.PGM,DATACLAS=DCLAS02,DISP=(NEW,KEEP),
//            LRECL=256,EXPDT=1996/033
```

In the example, the logical record length of 256 and the expiration date of February 2, 1996, override the corresponding attributes defined in the data class for the data set.

With SMS, you can associate a data class with any **new** data set, (whether or not it is system-managed). If you do not specify one or more of the DD parameters listed earlier (RECORG, RECFM, LRECL, KEYLEN, and so forth), the system uses the defined data class attributes.

For an existing system-managed DASD data set, note that you cannot use the volume-count subparameter to override the current volume count. (If you use the subparameter, the system ignores your specification and uses the current volume count.)

Overriding Attributes Defined in the Management Class

There are no attributes in the management class that you can specify via JCL. The storage administrator at your installation controls the migration and backup of SMS-managed data sets.

Overriding Attributes Defined in the Storage Class

You can specify volume serial numbers on the VOLUME parameter if the storage administrator has specified GUARANTEED_SPACE=YES in the storage class for the data set.

For example:

```
//SMSDS4  DD  DSNAME=MYDS4.PGM,STORCLAS=SCLAS04,DISP=(NEW,KEEP),
//          VOLUME=SER=(222333,333444)
```

In the example, the data set will reside on volume serials 222333 and 333444.

Protecting Data Sets with RACF

In many cases, your RACF user/group default data set profile is sufficient for the data sets you create.

However, you can override the default profile by coding the SECMODEL parameter. On the SECMODEL parameter, specify the name of an existing RACF data set profile.

For example:

```
//SMSDS5  DD  DSNAME=MYDS5.PGM,SECMODEL=(GROUP1.PGM),DISP=(NEW,KEEP)
```

Modeling Data Set Attributes

For a new data set, use the LIKE or REFDD parameter to copy to the new data set the attributes of an existing cataloged data set or a previously defined data set.

For example:

```
//SMSDS6  DD  DSNAME=MYDS6.PGM,LIKE=MYDSCAT.PGM,DISP=(NEW,KEEP)
    or
//SMSDS7  DD  DSNAME=MYDS7.PGM,DATACLAS=DCLAS02,DISP=(NEW,KEEP)
//SMSDS8  DD  DSNAME=MYDS8.PGM,REFDD=*.SMSDS7,DISP=(NEW,KEEP),
//          LRECL=1024
```

For both LIKE and REFDD, you can override data class attributes obtained from the referenced data set by coding those DD parameters that can be used to override attributes in these classes.

Index

Special Characters

//*DATASET control statement
use 12-6
//*ENDDATASET control statement
use 12-6
//*ENDPROCESS control statement
use 10-18
//*OPERATOR control statement
use 7-2
//*PROCESS control statement
use 10-18
when requesting processor 9-11
/*DEL control statement
use 24-6
/*EOF control statement
use 24-6
/*MESSAGE control statement
use 7-2
/*NOTIFY control statement
use 7-5
/*PRIORITY control statement
use 11-2
/*PURGE control statement
use 24-6
/*ROUTE control statement
use 24-1
/*SCAN control statement
use 24-6
/*SETUP control statement
use 15-52
/*SIGNOFF control statement
use 6-4
/*SIGNON control statement
use 6-4
*****
use in identifying in-stream data set 12-5

Numerics

3203 Printer Model 5
for printing sysout data set 25-2
3211 Printer
for printing sysout data set 25-4
3450 Diskette Input/Output Unit
identifying output data set 18-2
input data set 12-7
3800 Printing Subsystem
for printing high-density dump 25-5
for printing sysout data set 22-11, 25-2

A

abnormal termination
See termination
ACB (access method control block)
values for data set processing 13-6
ACCODE parameter
use 14-2
accounting information
to identify account 4-3
ACMAIN parameter
use 7-5
ACS routine
data set attribute description 13-6
use 13-7, 15-2, 15-17, B-3, D-2
with temporary data set 12-3
affinity
for multivolume data set 15-36
unit 15-30
explicit 15-30
implied 15-31
to subsystem data set 16-5
when requesting extended data set 15-35
unit and volume chart 15-34
unit and volume interaction 15-34
volume 15-21, 15-30
explicit 15-22
implicit 15-22
aging
See priority
allocation
chart 15-1
description 15-1, 15-53
dynamic 15-53
example 15-53
interaction 15-24
of device 15-2
affected by device status 15-3
concurrent 15-5
example 15-8
in JES3 system 15-12
number allocated 15-6
requesting more than one 15-6
of direct access space 15-44
example 15-48, 15-49
of tape or direct access volume 15-16
example 15-8
for indexed sequential data set if error occurs A-4
of virtual I/O 15-49
example 15-50, 15-51
with deferred volume mounting 15-6
example 15-6

© Copyright IBM Corp. 1988, 1999

allocation *(continued)*
with volume premounting 15-52
example 15-52
AMP parameter
use 13-6, C-4
automatic class selection routine
See ACS routine
average record
See AVGREC parameter
AVGREC parameter
use 13-7, 15-44, 15-45, D-3

B

BCP (base control program)
in relation to JCL statement 3-1
BSC (binary synchronous communication)
use 6-3, 6-4
BURST parameter
use 25-2
BYTES parameter
to limit job's output 7-4
in APPC scheduling environment 7-4, 10-13
in non-APPC scheduling environment 7-4, 10-14
use 10-19, 26-1

C

CANCEL subparameter
to cancel job that exceeds output limit 10-13
use 26-2
CARDS parameter
to limit job's output 7-4
in APPC scheduling environment 7-4, 10-13
in non-APPC scheduling environment 7-4, 10-14
use 10-19, 26-1
cards subparameter
use 26-1
catalog
in JES3 allocation 15-13
private
of data set 12-7
system
of data set 12-7
volume for
allocation and unallocation 12-7
cataloged procedure
See procedure
cataloged procedures
See procedure, cataloged and in-stream
cataloging
data set
not performed as requested 17-5
request 17-4
unsuccessful 17-5
use 17-4
when cataloged data set updated 17-4
character-arrangement table
dynamic selection 25-3
modifying 25-3
specification 25-3
CHARS parameter
use 25-2, 25-5
checkpointing
job execution 5-2
multivolume data set 16-4
submitting a job for restart B-10
sysout data set 22-10
CHKPT macro
restart control 5-2
CHKPT parameter
use 16-4
CHNSIZE parameter
use 24-1
CKPTLINE parameter
use 22-10
CKPTPAGE parameter
use 22-10
CKPTSEC parameter
use 22-10
class
job
description 11-1
for copying job 6-2
in holding job 6-1
to control performance 11-1
output
assigning data set 18-2
printing 22-5
use 22-5
CLASS parameter
for copying job 6-2
in holding job 6-1
to suppress sysout output 22-10
use 7-8, 10-17, 11-2, 18-2, 22-8
when requesting processor 9-11
CNTL parameter
use 16-5
CNTL statement
use 16-5
code
See return code
command statement
use 7-1, 7-2
comment
use 7-2
communication
chart 7-1
description 7-1, 7-9
from functional subsystem to programmer 7-6
example 7-6
from JCL to operator 7-2
example 7-2

communication *(continued)*
from JCL to program 7-3
example 7-3
from JCL to programmer 7-2
example 7-2
from JCL to system 7-1
example 7-2
from system to operator 7-3
example 7-4, 7-5
from system to TSO/E userid 7-5
example 7-6
from TSO/E userid to system 7-6
example 7-6
through job log 7-6
example 7-7, 7-9
COMPACT parameter
use 24-1
concatenation
data set 12-5
COND parameter
example 10-8
relationship of JOB and EXEC statement COND parameter 10-7
to force step execution 10-7
use 10-6
constructs
with SMS
description D-1
control
data set 13-1
CONTROL parameter
use 25-1
COPIES parameter
use 25-1, 25-2, 25-5
copies subparameter
use 25-1
copy
input stream 6-2
of data set name 12-5

D

data class
See also DATACLAS parameter
description D-1
overriding attribute D-3
DATA parameter
use in identifying in-stream data set 12-5
data set
cataloged
deleting 17-3
generation data set 17-5
placing in catalog 17-4
removing from catalog 17-5
specifying CATLG disposition 17-4
unit and volume information 15-25, 15-29
volume reference 15-25, 15-29
data set *(continued)*
concatenation 12-5
dummy
effect 16-1
nullification 16-2
to suppress sysout output 22-9
use 16-1
exclusive use 13-2
held
release 22-9
request 22-8
multivolume 15-18, 15-22
allocation consideration 15-18, 15-22
checkpointing 16-4
requesting space 15-46
passed
demounting volume 17-14
effect on volume retention 17-13
unit and volume information 15-25, 15-29
permanent 12-1
postponed definition 16-2
concatenation 16-2
reference 16-2
use 16-3
requesting resource 3-2
securing control 13-1
shared use 13-2
stacking 15-39
sysout
grouping 22-6
size 22-5
system-managed
description 7-6
temporary 12-3
deleting 17-3
use of VIO 15-49
with SMS D-1
data-set-sequence-number
use 12-10
DATACLAS parameter
use 13-6, B-2, B-3, D-2
DCB (data control block)
values for data set processing 13-5
values for sysout data set processing 19-1
values from cataloged data set 13-5
values from earlier DD statement 13-5
DCB parameter
use 13-5, 19-1, 25-2, 25-3, 25-5, A-3, A-6, B-6, B-9
DDNAME parameter
use 16-2, D-2
deadline
execution by 5-4
DEADLINE parameter
in deadline scheduling 5-4
in periodic scheduling 5-4

deallocation
See unallocation
default unit D-2
DEFER subparameter
use 15-6
deferred volume mounting
specification 15-6
DEFINE command B-2
deletion
not performed when data set uncataloged 17-5
of cataloged data set 17-3
of data set with unexpired expiration date 17-3
of generation data set B-9
of temporary data set 17-3
request 17-3
delimiter statement
use 12-5
dependency
before step execution
external 5-5
dependent job net
testing 5-5
use 5-5
description task for requesting data set resources
chart 13-1
description 13-1, 13-8
of data attribute 13-5
of data attributes
example 13-6, 13-7, 13-8
of status 13-1
example 13-2
description task for requesting sysout resources
chart 19-1
description 19-1
of data attribute 19-1
example 19-1
DEST parameter
use 24-1
destination
default 24-4
multiple 24-2
destination control task for requesting sysout resources
chart 24-1
description 24-1, 24-8
to another processor 24-6
example 24-6
to assist in sysout distribution 24-8
to internal reader 24-6
example 24-7
to local or remote device or to another node 24-1
example 24-4, 24-5
in JES2 network 24-2
in JES3 network 24-5
to terminal 24-7
example 24-8
device
allocation 15-2
management in JES3 system 15-12
number allocated 15-6
specifying as destination for sysout data set 24-1
DFSMSdfp
with SMS-managed data set D-1
directory
of PDS 9-1
DISP parameter
use 12-7, 13-1, 17-2, A-3, A-6, B-3, B-6, B-8, B-9, D-2
when data set is cataloged 12-7
DJC (dependent job control)
use 5-5
DLM parameter
use 12-5
documentation
job and its resource requirement 7-2
DPAGELBL parameter
use 20-1
DSID parameter
use 12-7, 18-2
DSNAME parameter
use 12-1, 12-5, 15-19, 18-1, A-2, A-6, B-3, B-5, B-7, D-2
DSNTYPE parameter
use 13-7, D-3
DUMMY parameter
use 16-1, 22-9
with SUBSYS parameter 16-4
dump
after error 10-19
format 25-5
high-density 10-19
DUMP subparameter
use 26-2
dynamic
allocation 15-53
unallocation 17-1
DYNAMNBR parameter
use 15-53

E

end processing task for requesting data set resource
chart 17-1
description 17-1, 17-14
disposition of data set 17-1
bypassing 17-9
cataloging 17-4
default 17-9
deleting 17-3
effect of device type 17-2
example 17-10, 17-12
keeping 17-3

end processing task for requesting data set resource *(continued)*
disposition of data set *(continued)*
passing 17-5
uncataloging 17-5
when no abnormal termination disposition coded 17-2
disposition of volume 17-12
example 17-13, 17-14
of removable volume 17-13
release of unused direct access space 17-12
example 17-12
unallocation 17-1
example 17-1
end processing task for requesting sysout resource
chart 23-1
description 23-1
unallocating 23-1
example 23-1
entering jobs
task in job control 2-5
error
scanning JCL 10-17
EVEN subparameter
example 10-13
to force step execution 10-7
event
external
holding job 6-1
execution
at remote node 5-7
considerations for 5-8
example 5-8
chart 5-1
deadline or periodic 5-4
example 5-4
use 5-4
description 5-1, 5-8
of procedure 5-2
example 5-2
of program 5-1
example 5-1
when dependent on other job 5-5
when dependent on other jobs
example 5-5
when restarting and with checkpointing 5-2
example 5-3
EXPDT parameter
use 13-7, 17-11
expiration date
for data set
deleting prior to date 17-11
effect on disposition 17-11
request 17-11
when unexpired 17-3
extent
in allocation of direct access space 15-46
external writer
See writer

F

FAILURE parameter
in restart 5-3
FCB parameter
use 25-1, 25-5
FETCH parameter
use 7-3
FLASH parameter
use 25-2
FORMDEF parameter
use 22-11
FORMS parameter
use 25-5
forms subparameter
use 25-1
FREE parameter
use 17-1, 23-1

G

GDG (generation data group)
building base entry B-2
cataloging data set 17-5
data set label B-4
defining attribute B-4
identifying 12-2
type of data set B-2
generation data set
creating B-3, B-5
deleting and uncataloging B-9
description B-1, B-13
example B-10
retrieving B-2, B-7
rules when submitting job for restart B-10
graphic character modification modules
modifying 25-3
GROUP parameter
use 8-1
GROUPID parameter
use 22-6
grouping
sysout data sets
demand setup 22-6
request 22-6
subgroup 22-6
guaranteed space D-4
with system-managed DASD data set 15-17

H

hard-copy log
description 7-7
high-density dumps
request 25-5
HOLD parameter
in holding job 6-1
use 22-8
holding
job entrance 6-1
of sysout data set 22-8
releasing 22-9
use 22-8

I

I/O-to-processing ratio
use 11-4
identification task for entering jobs
chart 4-1
description 4-1, 4-5
of account 4-3
example for local execution 4-4
example for remote execution 4-4
for local execution 4-3
for remote execution 4-4
of job 4-1
example 4-1
of procedure 4-2, 4-3
example 4-2
of programmer 4-4
example 4-5
of step 4-2
example 4-2
identification task for requesting data set resources
as TCAM message data set 12-10
example 12-11
by location on tape 12-10
example 12-10
chart 12-1
description 12-1, 12-11
from or to terminal 12-11
example 12-11
of data set 12-1
example for generation data set 12-3
example for indexed sequential data set 12-3, 12-5
example for partitioned data set 12-2, 12-4
example for permanent data set 12-2
example for temporary data set 12-4
example when copying data set name 12-5
of data set on 3450 Diskette Input/Output Unit 12-7
example 12-7
of in-stream data set 12-5
example 12-6, 12-7
identification task for requesting data set resources *(continued)*
through catalog 12-7
example 12-8
through label 12-8
example 12-10
identification, task for requesting sysout resources
as a sysout data set 18-1
example 18-1
chart 18-1
description 18-1, 18-2
of data set on 3450 Diskette Input/Output Unit 18-2
example 18-2
of output class 18-2
example 18-2
IEBIMAGE utility program
use for character-arrangement table 25-3
use in updating SYS1.IMAGELIB 25-3
IEBUPDTE utility program
use 9-5
IEFBR14 program
considerations when using 10-18
description 10-17
use in testing 10-17
IEHPROGM utility program
use B-9
IF/THEN/ELSE/ENDIF statement construct
example 10-5
level of evaluation
job level 10-3
step level 10-4
to force step execution 10-4
use 10-1
with COND parameter 10-4
IN subparameter
use 14-3, 14-4
in-stream
See input stream
in-stream procedure
See procedure, cataloged and in-stream
in-stream procedures
See procedure
INCLUDE group
specifying library containing 9-6
INCLUDE statement
example 4-3
use 4-3
independent mode
processor
requesting 9-10
index area
description A-1
identifying 12-3, 12-4
INDEX parameter
use 12-3, 25-4

indexed sequential data set
area arrangement A-5
creating A-1
description A-1, A-8
example A-7
identifying 12-3, 12-4
parameters for retrieval or extension A-6
retrieving A-5
specific track request 15-49
system assigned space request 15-48
when allocation error occurs A-4
indexing
of sysout data set margin 25-4
input control task for entering jobs
by copying input stream 6-2
example 6-3
by holding job entrance 6-1
example 6-2
use 6-1
by holding local input reader 6-2
example 6-2
chart 6-1
description 6-1, 6-4
from remote work station 6-3
input stream
definition 2-13
description 3-1
device 2-13
example 2-13
identification of data set 12-5
integrity processing
chart 13-4
definition 13-1
for other than permanent data set 13-3
for permanent data set 13-3
of data set 13-2
interpretation
punched sysout data set
request 25-5
INTRDR subparameter
use 24-6
invalid syntax
scanning for 10-17
IORATE parameter
use 11-4
ISMF (Interactive Storage Management Facility)
use 13-6, 13-8, 15-2, 15-17, D-2

J

JCLLIB statement
specifying library 9-6
for INCLUDE group 9-6
JCLTEST subparameter
use 10-17
JESDS parameter
use 7-7, 7-8
job
background
defining 7-5
batch
defining 7-5
control 2-1
description 3-1
entering 2-5, 3-1
example 2-5
jobstep example 2-5
predecessor 5-5
processing 3-2
requesting resource 2-3
successor 5-5
termination
when data set cannot be cataloged 17-5
job input control
See input control task for entering jobs
job log
description 7-6
execution time messages in log 10-15
for communication from JCL to programmer 7-2
output class 7-7
printing 7-7
with sysout data set 7-8
JOBCAT catalog
use 12-8
JOBLIB
See library, private
jobname
to identify job 4-1
JOURNAL parameter
in restart 5-2
JSTTEST subparameter
use 10-17

K

keeping
data set
request 17-3
when data set uncataloged 17-5
disposition
for tape volume 17-13
key length
See KEYLEN parameter
key offset
See KEYOFF parameter
KEYLEN parameter
use 13-7, D-3
KEYOFF parameter
use 13-7, D-3

L

label
- data set
  - for cataloged or passed data set 12-9
  - for nonspecific volume request 12-9
  - for specific volume request 12-10
  - use 12-8

LABEL parameter
- use 12-8, 17-11, A-3, B-6, B-9

library
- defining 9-1
- private 5-1, 9-1
  - adding 9-2
  - concatenating 9-3
  - creating 9-2
  - JOBLIB statement 5-1, 9-1
  - retrieval 9-3
  - STEPLIB statement 5-1, 9-1
  - use 9-2
- procedure 9-5
  - updating 9-5
  - use for procedure 3-1
- residence for executable program 5-1
- SYS1.IMAGELIB
  - use 22-11
  - use in copy modification 25-3
- system 5-1, 9-1
  - SYS1.LINKLIB 5-1, 9-1
  - use 9-1
- temporary 5-1, 9-1
  - creating 9-4
  - use 9-4

LIKE parameter
- use 13-7, B-3, D-4

limit
- sysout output
  - request 26-1
  - use 26-1
  - when exceeded 26-2

LINDEX parameter
- use 25-4

LINECT parameter
- use 25-1

linect subparameter
- use 25-1

LINES parameter
- to limit job's output 7-4
  - in APPC scheduling environment 7-4, 10-13
  - in non-APPC scheduling environment 7-4, 10-14
- use 10-19, 26-1

lines subparameter
- use 26-1

location
- of data set on tape 12-10

log
- *See* hard-copy log
- *See* job log

log subparameter
- use 7-7

LOGOFF command
- use 6-4

LOGON command
- use 6-4

loop
- program
  - stopping execution 10-15

LRECL parameter
- use 13-7, D-3

LREGION parameter
- use 9-9

M

management class
- *See also* MGMTCLAS parameter
- description D-1
- overriding attribute D-3

member
- in PDS 9-1

message
- during volume mounting 7-3
- from system for job 7-7
- when job exceeds output limit 7-4

MGMTCLAS parameter
- use 13-7, 13-8, D-2

migration and backup D-3
- description 13-7
- with SMS 13-7

mode
- process
  - requesting for sysout data set 22-7

model
- data set attributes
  - summary for SMS-managed data set D-4
  - with SMS 13-7

modification
- for sysout data set 25-3

MODIFY parameter
- use 25-2

mount
- volume
  - deferred 15-6
  - message control 7-3
  - premounting 15-52

MSGCLASS parameter
- controlling copied input stream 6-2
- use 7-7, 7-8, 18-2, 22-5, 24-6

MSGLEVEL parameter
- use 7-7
  - use in controlling job log listing 7-2

N

naming
data set 12-1
temporary data set 12-3
net
See dependent job net
node
See remote node
NOLOG parameter
use 7-7
NOPWREAD subparameter
use 14-3
notification
of TSO/E userid 7-5
NOTIFY parameter
use 7-5
null statement
example 4-1
to identify job end 4-1
NULLFILE subparameter
use 16-1
nullification
of dummy data set 16-2
of dummy status for sysout data set 22-9

O

ONLY subparameter
example 10-13
to force step execution 10-7
operating system
content 3-1
ORG parameter
use 24-1
OUT subparameter
processing with 14-4
use 14-3
OUTDISP parameter
of OUTPUT JCL statement
to hold a sysout data set 22-8
to suppress output 22-10
OUTLIM parameter
use 24-6, 26-1
output formatting
chart 25-1
description 25-1, 25-5
of dump on 3800 Printing Subsystem 25-5
of dumps on 3800 Printing Subsystem
example 25-5
to 3211 Printer with indexing feature 25-4
example 25-4
to 3800 Printing Subsystem 25-2
example 25-4
to any printer 25-1
example 25-2
output formatting *(continued)*
to punch 25-5
example 25-5
OUTPUT JCL statement
adding parameter 22-2
changing //*FORMAT statement 22-4
changing /*OUTPUT statement 22-4
references to multiple statements 22-2
use 22-1
output limiting
chart 26-1
description 26-1, 26-3
example 26-2
in a non-APPC scheduling environment 26-2
in an APPC scheduling environment 26-1
messages when limit exceeded 7-4
request 26-1
terminating job when limit exceeded 10-13
use 26-1
when exceeded 26-2
OUTPUT parameter
use 22-1
overflow area
description A-1
identifying 12-3, 12-4
OVFLOW subparameter
use 12-3

P

PAGEDEF parameter
use 22-11
PAGES parameter
to limit job's output 7-4
in APPC scheduling environment 7-4, 10-13
in non-APPC scheduling environment 7-4, 10-14
use 10-19, 26-1
parallel mounting
of volumes
to request more than one device 15-6
PARM parameter
use in communicating from JCL to program 7-3
values for IBM-supplied program 7-3
partition
See spool partitions
pass
data set
demounting of volume 17-14
disposition when data set unreceived 17-6
effect on volume retention 17-13
receiving passed data set 17-5
requesting 17-5
when step abnormally terminates during execution 17-6
PASSWORD parameter
use 8-1, 14-3

passwords
in protecting data set 14-3
PDS (partitioned data set)
identifying 12-2, 12-4
member 12-2, 12-4
use as library 9-1
PDSE (partitioned data set extended)
member 12-2, 12-4
PEND statement
to identify procedure end 4-2
PERFORM parameter
use 11-3
performance control task for processing jobs
by I/O-to-processing ratio 11-4
example 11-4
by job class assignment
by job class assignment 11-1
example 11-2
by performance group assignment 11-3
example 11-4
by selection priority 11-2
example 11-3
chart 11-1
description 11-1, 11-4
performance control task for requesting sysout resources
by queue selection 21-1
example 21-1
chart 21-1
description 21-1
performance group
use 11-3
periodic
execution 5-4
PIMSG parameter
use 7-6
postponing
specification of data set 16-2
prime area
description A-1
identifying 12-3, 12-4
PRIME subparameter
use 12-3
print
controlling format 25-1
protecting printed output 20-1
sysout data set
on same listing 22-5
scheduling 23-1
simultaneously on different printer 22-6
printed output
protection 20-1
priority
aging 11-3
not useful in controlling execution order 6-2
selection 11-2
for sysout data set 21-1
priority *(continued)*
selection *(continued)*
ignoring 21-1
use 11-2
PROC parameter
to execute procedure 5-2
use 9-5
PROC statement
to identify procedure 4-2
procedure
cataloged and in-stream
description 3-1
execution 5-2
overriding DD statement 15-25, 15-29
testing 3-1
in private library 9-5
processing
nonstandard
definition 10-18
use in testing 10-18
processing control task for processing jobs
by conditional execution 10-1, 10-13
example when return codes tested 10-8
in APPC scheduling environment 10-14
in non-APPC scheduling environment 10-14
by timing execution 10-15
example 10-15, 10-16
chart 10-1
description 10-1, 10-20
for testing 10-16
by altering usual processing 10-16
by dumping after error 10-19
example when dumping 10-19
example when scanning JCL 10-17
example when using IEFBR14 10-18
example when using nonstandard processing 10-18
processing control task for requesting data set resources
by postponing specification 16-2
example 16-3
by subsystem 16-4
example 16-5
by suppressing processing 16-1
example 16-2
by TCAM job or task 16-5
example 16-5
chart 16-1
description 16-1, 16-5
with checkpointing 16-4
example 16-4
processing control task for requesting sysout resources
by checkpointing 22-10
by external writer 22-7
example 22-7

processing control task for requesting sysout resources *(continued)*
by holding 22-8
example 22-9
by mode 22-7
example 22-7
by PSF 22-11
example 22-11
by segmenting 22-5
by suppressing output 22-9
example 22-10
chart 22-1
description 22-1, 22-12
with additional parameter 22-1
with additional parameters
example 22-2
with checkpointing
example 22-11
with other data set 22-5
with other data sets
example 22-5, 22-6
processor
as output destination 24-6
selecting in JES2 9-10
selecting in JES3 9-10
selecting using a scheduling environment 9-9
PROCLIB parameter
use 9-5
programmer's name
to identify 4-4
PROTECT parameter
use 14-1
protection task for entering jobs
chart 8-1
description 8-1
through RACF 8-1
example 8-1
protection task for requesting data set resources
by password 14-3
by passwords
example 14-3
chart 14-1
description 14-1, 14-4
for ISO/ANSI/FIPS Version 3 tape 14-2
for ISO/ANSI/FIPS Version 3 tapes
example 14-2
for SMS-managed data sets
description 14-2
summary D-4
of access to BSAM and BDAM data set 14-3
of access to BSAM and BDAM data sets
chart 14-3
example 14-4
through RACF 14-1
example 14-2
protection task for requesting sysout data set resources
chart 20-1
description 20-1
example 20-1
with RACF 20-1
PRTY parameter
use 11-2, 21-1
PSF (Print Services Facility)
control 22-11
punch
sysout data set
formatting 25-5
interpretation 25-5
scheduling 23-1

Q

QNAME parameter
use 12-10, 16-5

R

RACF (Resource Access Control Facility)
data set protection 14-1
protecting printed output 20-1
protection through 8-1
with in-stream data set 12-6
with sysout data set 18-1
RD parameter
in restart 5-2
reader
input
holding 6-2
internal
as output destination 24-6
description 3-1
limiting record 24-6
message class 24-6
sending directly to JES 24-6
receive
passed data set
requesting 17-5
RECFM parameter
use 13-7, D-3
record organization
See RECORG parameter
RECORG parameter
use 13-7, D-3
REFDD parameter
use 13-7, B-3, D-4
reference DD statement
See REFDD parameter
relative generation numbers
definition B-1

release
held sysout data set
requesting 22-9
remote node
execution 5-7
specifying as destination for sysout data set 24-1
remote terminal
use 6-3
remote work station
use 6-3
requesting resources 3-6
for data set 2-3
task in job control 2-3
tasks 3-6
task chart 3-6
resource control task for entering jobs
chart 9-1
description 9-1, 9-13
of address space 9-6
example 9-8, 9-9
of INCLUDE group 9-6
of procedure library 9-5
example 9-5
of processor 9-9
example 9-10, 9-11
of program library 9-1
creating and adding example 9-3
example of concatenating 9-4
example of retrieving 9-3
example of temporary 9-4
of spool partition 9-11
example 9-12
restart
after abnormal termination 5-2
after JES2 system failure 5-3
after JES3 system failure 5-3
automatic checkpoint 5-2
automatic step 5-2
deferred checkpoint 5-2
deferred step 5-2
use 5-3
when job contains generation data set B-10
RESTART parameter
in restart 5-2, 5-3
RETAIN subparameter
use 17-12, 17-13
retention
of tape volume
demounting of volume 17-14
requesting 17-13
use 17-13
RETPD parameter
use 13-7, 17-11
return code
compatible test 10-3, 10-8
with COND parameter 10-8
with IF/THEN/ELSE/ENDIF statement
construct 10-3
return code *(continued)*
example 10-8
testing 10-2, 10-6
with COND parameter 10-6
with IF/THEN/ELSE/ENDIF statement
construct 10-2
RJE (remote job entry)
use 5-7, 6-3
RJP (remote job processing)
output destination 24-5
use 5-7, 6-4

S

scanning
syntax for error 6-2, 10-17
scheduling environment 9-9
SCHENV parameter
of JOB statement 9-9
example 9-10
use 9-9
scratch disposition
for tape volume 17-13
SECLABEL parameter
use 8-1
SECMODEL parameter
use 14-2, D-4
security model
See SECMODEL parameter
SEGMENT parameter
of DD statement
use 22-5
setup
of devices
altering 15-16
explicit 15-15
high watermark 15-14
in JES3 system 15-13
job 15-13
SETUP parameter
mount messages for volume 7-3
use 15-12
SMF (System Management Facilities)
to establish exit routine when execution time exceeded 10-15
SMS (Storage Management Subsystem)
description D-1
SMS-managed data set
attribute description 13-6
construct D-1
creating a generation data set B-3
definition D-1
generation data set B-1, B-2
migration and backup 13-7, D-3
modeling attribute 13-7
multivolume data set 15-18
protection 14-2, D-4

SMS (Storage Management Subsystem) *(continued)*
SMS-managed data set *(continued)*
retrieving a generation data set B-7
space allocation 15-44
specifying expiration date 17-11
specifying retention period 17-11
specifying volume serial 15-17
unit allocation 15-2
volume allocation 15-17
with DATACLAS DD parameter 13-6
with DD VOLUME=REF subparameter 15-17
with VIO data set 15-49
specifying directory record 15-47
with private catalog 12-8
with temporary data set 12-3
with VSAM data set C-1
SNA/SDLC (systems network architecture synchronous data link control)
use 6-3, 6-4
space
allocation
primary 15-45
secondary 15-46, 15-47
releasing when unused 17-12
request
block 15-44
cylinder 15-45
for indexed sequential data set 15-48
for PDS directory 15-47
record 15-45
specific track 15-49
system assignment 15-44
track 15-45
with user label 15-46, 15-49
SPACE parameter
use 13-7, 15-44, 17-12, A-4, B-6, D-3
SPART parameter
use 9-12
spinning off
sysout data set
request 23-1
use 23-1
spool partitions
controlling allocation 9-11
stacking, data set 15-39
statements
See also JCL, JES2, and JES3
purpose 1-1, 1-3
station
See remote work station
status
of data set
specifying 13-1
of device
affect on allocation 15-3
step
description 3-1
maximum number 3-2
STEPCAT catalog
use 12-8
STEPLIB
See library, private
stepname
to identify step 4-2
storage
administrator D-2
central 9-7
region size 9-8
class
overriding attribute D-4
summary D-1
logical 9-9
real 9-7
region size 9-8
requesting 9-7
virtual 9-7
region size 9-7
storage class
See STORCLAS parameter
STORCLAS parameter
use 15-2, 15-17, B-2, B-3, D-2
with temporary data set 12-3
SUBSYS parameter
use 16-4
subsystem
printing message 7-6
program control statement 16-5
request 16-4
suppression
of sysout output
request 22-9
using OUTPUT JCL statement 22-10
with started task 22-10
syntax
scanning
for error 6-2, 10-17
SYS1.LINKLIB
See library, system
SYS1.PROCLIB
See catalog
SYS1.PROCLIB system procedure library
use for procedure 3-1
SYSABEND statement
use 10-19
SYSAFF parameter
use 9-10
SYSAREA parameter
use 20-1
SYSCHK DD statement
in restart 5-2

SYSCKEOV DD statement
use 16-4
SYSMDUMP statement
use 10-19
sysout data set
printing with job log 7-8
SYSOUT parameter
use 7-8, 18-1, 18-2
SYSTEM parameter
use 9-11
system-generated qualified name
for temporary data set 12-3
SYSUDUMP statement
use 10-19

T

task
chart 3-2
description 3-1
for entering jobs
chart 3-3
for processing jobs
chart 3-5
for requesting sysout data set resources
chart 3-8
TCAM (telecommunications access method)
message data set 12-10
processing of TCAM message data set 16-5
temporary data set 12-3
TERM parameter
use 12-11, 24-7
terminal
See also remote terminal
as output destination 24-7
identifying data set 12-11
termination
abnormal
data set disposition 17-2
disposition of unreceived passed data set 17-7
effect on disposition 17-2
effect on passing of data set 17-6
execution time exceeded 10-15
forcing execution of later step 10-4, 10-7
output limit exceeded 10-13
normal
data set disposition 17-2
restarting 5-2
when system cannot catalog data set 17-5
THRESHLD parameter
use 22-5
TIME parameter
use 10-15, 10-16
TRC parameter
use 25-2, 25-3
TSO/E (time sharing option) userid
as output destination 24-6
notifying when job complete 7-5
RACF protection parameters from logon 8-1
TYPE parameter
when requesting processor 9-11
TYPRUN parameter
copying job 6-2
holding job 6-1
use 10-17

U

UCS parameter
use 25-1, 25-3
unallocation
of data set, volume, and device 17-1
sysout data set 23-1
uncatalog
data set
generation data set B-9
request 17-5
unit
See device
UNIT parameter
definition during system initialization 15-5
for output data set 15-5
relationship to VOLUME parameter 15-24, 15-29
use 15-2, A-2, A-6, B-6, B-8
when requesting processor 9-11
unreceived data set
at abnormal termination 17-7
at end of job 17-8
disposition 17-6
UPDATE parameter
in holding job 6-1
use 9-5
USER parameter
use 8-1
use in identifying job with TSO/E userid 7-6

V

VIO (virtual input/output)
backward reference 15-51
use 15-49
volume
attribute
affect on device allocation 15-35
assigning 15-21
definition 15-18
permanently resident 15-19
private 15-18
public 15-18
removable 15-19
reserved 15-19
retention 17-13

volume *(continued)*
attribute *(continued)*
storage 15-18
VOLUME parameter
referencing in earlier DD statement 15-25, 15-29
relationship to UNIT parameter 15-24, 15-29
use 13-7, 15-17, 15-18, 15-19, 15-20, A-2, A-6, B-6, B-8, D-3
volume requests
nonspecific 15-18, 15-20
allocation 15-20
label type 12-9
number per DD statement 15-23
specific 15-19
allocation 15-19
label type 12-10
VSAM (virtual storage access method)
data set
creating C-1, C-4
description C-1, C-7
parameter C-2
parameters C-4
parameters to avoid C-6, C-7
retrieving C-1, C-4

W

WARNING subparameter
to send warning message when job exceeds output limit 7-4
use 26-2
work station
See remote work station
writer
external
for processing sysout data set 22-7
request 22-7
WRITER parameter
use 22-7

X

XMIT JCL statement
with JES3 5-8

Communicating Your Comments to IBM

OS/390
MVS JCL User's Guide

Publication No. GC28-1758-05

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  1-(914)-432-9405
- If you prefer to send comments electronically, use this network ID:

  mhvrcfs@us.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

OS/390
MVS JCL User's Guide
Publication No. GC28-1758-05

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
GC28-1758-05

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5647-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC28-1758-05